COMPUTATIONAL ANALYSIS OF MULTI-STOREY FRAMED DECK STRUCTURES

by

Derek C. Breeden

A Thesis

submitted to the University of Warwick

for the Degree of Doctor of Philosophy.

September 1974.

The author wishes to thank Mr. J. Wright for his advice and supervision of the work and Mr. D. Anderson and Mr. M. Larcombe for their help and encouragement. He also wishes to express his thanks to Miss P. Sotheran who prepared the typed manuscript and to the Science Research Council for their financial assistance.

ABSTRACT.

A new approach to the analysis of three-dimensional framed structures whose prismatic members are interconnected by slab floors is presented. The method is illustrated by the computer program described. The structures are restricted to those of an orthogonal nature with rigid, interconnected elements. The method of solution is a stiffness matrix approach utilising a finite element formulation for the in-plane and out-of-plane action of the floor deck. A process of condensation is applied to the stiffness matrices to reorganise and reduce storage demands. A 'part-structure' condensation is employed floor by floor, making the process highly suitable for multi-storey structures.

Several examples are given to illustrate the use of the program and to demonstrate its versatility. Also the wide range of results obtainable and the methods of presentation are given.

The program is used together with one for a plane frame analysis and one for a rigid, orthogonal, skeletal space frame analysis, both of which are described in this volume, in order to investigate and compare the solutions obtained for the test frames described. Some results are presented for the investigation into the stiffening effect of the floor slabs on the bare frame. Also the use of three-dimensional sub-frames for the analysis of beams in a space frame is presented.

The effect on the total structure behaviour of finite element mesh concentration for the floor representation is also demonstrated.

LIST OF CONTENTS.

Nomenclature.

A	- cross-sectional area; transformation matrix.
AX1 (2), AXL1(2)	- axial force at end 1(2).
D	- elasticity matrix; flexural rigidity.
DEFX(Y,Z)	- deflection in direction of X(Y,Z) axis.
E	- Young's Modulus.
$\underline{E}$	- strain vector.
$F_{x1(2)}$,FX1(2) etc	- force in X direction at end 1(2), similarly for Y,Z directions.
F_x, F_y	- in-plane forces on plate element.
G	- shear modulus.
I_y,I_z,IY,IZ	- moment of inertia about Y(Z) axis.
J	- polar moment of inertia.
K	- stiffness matrix.
L	- length of side of slab.
$M_{x1(2)}$,MX1(2) etc	- moment about X axis at end 1(2), similarly for Y,Z directions.
M_x,M_y,M_z	- moment about X, Y, Z axes.
M_x, M_y	- moment per unit length in X, Y direction.
M_{xy}	- twisting moment per unit length.
MOM1(2)	- moment at end 1(2).
N	- number of nodes.
P	- load vector; force vector.
$P_{x(y,z)}$	- load in X(Y,Z) direction.
ROTN	- rotation.
ROTX(Y,Z)	- rotation about X(Y,Z) axes.
SHR1(2)	- shear force at end 1(2).

SY1,SZ1,SY2,SZ2	- force in Y(Z) direction at end 1(2).
S_x, S_y	- direct stresses.
S_{xy}	- shear stress.
U	- uniformly distributed vertical load.
X,Y,Z	- member axes.
X',Y',Z'	- structure axes.
XC,YC,ZC	- X,Y,Z coordinates.
X-DIR, Y-DIR	- displacement in X, Y direction.
d	- displacement vector.
dx, dy, dz	- displacement in X,Y,Z direction.
n	- number of members.
q	- load per unit area.
t	- thickness of slab.
u	- deflection of plate in X direction.
v	- deflection of plate in Y direction.
w	- shape function; vertical deflection of plate.
$\theta x(y,z)$	- rotation about X(Y,Z) axes.
ν	- Poisson's ratio.

1. INTRODUCTION.

1.1 Historical and Critical Review of the Field of Research.

The elastic analysis of structures has become a highly complex and extensive region for research. It can range from element or material behaviour to a more macroscopic scale such as total structural behaviour of large multi-storey frames.

Before the advent of computers, analyses for large systems were tedious, being established using tabular methods suited to the drawing office and they could not be expected to handle complex structures. However, with the arrival of the computer, techniques utilising its attributes were extensively developed. Whereas an increase in the number of equations to be solved made hand solution undesirable, no loss of efficiency is apparent in a computer solution.

The mathematical statement of the structural analysis problem was expressed as a series of simultaneous equations in matrix notation, since this was a highly attractive form for the programming of machines. Classical theory was used to obtain equations relating the unknown joint displacements of the structure to the applied loads, the structural properties being expressed as stiffnesses[1]. It is not clear who first introduced the stiffness approach to structural analysis as it seems to have developed in various centres simultaneously. However, the enormous impact the technique had, and still has, on structural analysis since computers became available is clear. The method was first applied to simple plane frames and proved

to be far superior, when applied to more complex structures, compared with other methods available such as moment distribution.

Although computer programs were initially developed as a research tool the eventual aim was for a working design aid and therefore the programming of the technique for practising engineers became the main objective. Here, the efficiency of programs in continual use became important, and the programming approach became more refined with the influence of user requirements.

Initially a plane frame analysis was a straightforward programming exercise once the mathematical formulation had been derived, since generally the computer could handle the demands the analysis made on it especially when symmetry and bandwidth were utilised in the structure matrix. An early program was developed by Livesley[1] who, it appears, instigated the development in this direction on a wide scale and his book on 'Matrix Methods of Structural Analysis' is a sound exposé on the subject.

Both the stiffness and flexibility methods are said to have been developed in the 1860's by Maxwell and Clebsch. The stiffness or displacement method developed into the slope-deflection technique as applied to the solution of continuous beams, but little use was made of the method for many years due to the need to solve many simultaneous equations by hand. However, being well suited to modern computers, the stiffness approach was given a new lease of life with their intervention.

General purpose programs were soon available but at this stage were confined to plane frame analysis, which proved to be and still proves to be a very useful and readily implemented aid to the structural designer. Although only an idealisation of three-dimensional behaviour, for design purposes it has proved adequate. However, where a structure does not readily lend itself to simulation by a series of plane frames or the loading is not in the same plane as the frame, then the demand is for the next stage, namely space frame analysis and the first extension to a three-dimensional analysis was that for a skeletal space frame.

With the extension to three-dimensional space frames the demands on the computer store increased. In the plane frame case only three degrees of freedom at a node are required. Therefore for a frame of N nodes there are 3N equations to be solved. The corresponding structure matrix for this is 3N x 3N in size, but this is both symmetric and banded, which allows for a much more compact form of data storage. For a space frame, the number of unknowns increases to 6N and the storage requirements increase alarmingly. This was only a problem whilst the size of computers was small but with the development of better machines and suitable backing store the problem became surmountable, enabling three-dimensional skeletal analysis to be undertaken.

However, for more encompassing structural behaviour, research workers now aimed for what might be called 'the analysis of complete structures' where, for an elastic analysis,

the behaviour of a beam or column member could be defined by the slope-deflection equations. In order to include elements such as floors or walls into the analysis the elastic behaviour or stiffness of these elements would have to be defined. The development of sophisticated techniques such as finite difference, finite element[2], localised Ritz[3] and other variational methods allied to computers has enabled these extra elements to be incorporated. However, it can be here that the problems start.

The finite difference formulation for the analysis of thin plates can be extended to provide a force/displacement relationship as developed by Croll[4] or Salonen[5] which could be incorporated into a structure analysis, but results provided by this are inferior to those of the finite element method[2]. This technique enables the complete definition of all the required variables except one, that is the in-plane rotational stiffness of the plate element. Despite this, it does provide the best force/displacement relationships for ease of use in programming for a complete structure analysis.

Thus with the theoretical obstacles removed all would appear straightforward but, as stated earlier, it can be here that the real problems start to occur. In order to formulate the arithmetic into a systematic solution process for a computer program it is evident that the demands on computer storage can be excessive. Thus it is inevitable that computational techniques for reduction or simplification are required to temper these demands and, if necessary or

justifiable, this may be achieved by approximation.

Therefore a new direction evolved where approximate or pseudo-three-dimensional structural behaviour could be handled more easily than the full elastic analysis of a diaphragmed space frame. These approximating techniques are usually applied to structures whose nature is such that it diminishes any error in the approximation.

An early approach put forward by Clough[6] was one where the floors were assumed to act as rigid diaphragms and that torsion of the structure could be neglected. Here, horizontal deflection of frames and wall elements would be equal, which could result in substantial error. A similar procedure was used by Weaver and Nelson[7] in their paper on 'Three-dimensional analysis of tier buildings' where all joint displacements and all element stiffnesses are considered. However, again the floor slabs were assumed to be diaphragms, rigid in their own plane.

In a paper by Majid and Williamson[8] 'a method for the analysis of general complete structures consisting of a combination of one or more of three basic components of the structure, namely: prismatic members, plate elements subject to forces within the plane of the elements and plate elements subject to out-of-plane forces' is described together with the associated computer program. The process uses a matrix displacement method and consists of assembling a total stiffness matrix for the structure from individual element stiffness

matrices, where finite element techniques are used to develop the plate element stiffness.

The program described is very comprehensive, including the analysis of general frames in three-dimensions. However, it does require that all the degrees of freedom are represented in the structure stiffness matrix at the same time and because of this storage requirements can soon lead to problems with quite small structures, especially if diaphragm sections are split into many elements. The mathematical analysis is justified by experimental results given in the paper, and it is stated that the addition of a floor slab increases by about 25% the in-plane sway stiffness of the bare frame.

The paper by Bond[9] describes a similar analysis and goes on to give information of a program for studying the design of reinforced concrete structures supported on columns. In order to reduce the complexity of the problem, the number of degrees of freedom has been reduced from the 6 per node in a space frame structure. Here only the flexural behaviour of the slab element has been considered and torsional resistance of the columns has been omitted. An approximation is used to overcome the lack of continuity between beam and slab elements, where beam elements are restricted to only three degrees of freedom.

To take the approximation one stage further, Stamato and Stafford Smith[10] give an 'Approximate method for the three-dimensional analysis of tall buildings'. The technique consists

of analysing two-dimensional panels arranged orthogonally or obliquely, whose compatibility of displacements is provided for at panel intersections. Here a two-dimensional approximation to the three-dimensional situation is undertaken with the assumption that certain degrees of freedom in these cases are not significant. Heavy restrictions are put on the floor behaviour, where elements are treated as rigid diaphragms of infinite in-plane and zero transverse stiffness. However, the method does point to the use of repetitive elements or sections of structure throughout the space frame.

A different approach to the problem is put forward by Zienkiewicz, Parekh and Teply[11] in their paper on 'buildings composed of floor and wall panels', where the primary load bearing action is taken by in-plane forces. The paper demomstrates this approach by examples and expresses the economy in the adoption of this form of action. However, it only corresponds to those structures of the panel nature described, e.g. load bearing shear wall structures etc.. Thus it is not applicable to column/beam/slab structures where the bending action is more predominant.

An approximation of structural behaviour, illustrated in a paper by Majid and Croxton[12], enables the 'wind analysis of complete building structures by influence coefficients'. Conclusions are made that there is a real need for special purpose programs for the various kinds of three-dimensional analysis, and also that 'the assumption that the slabs are rigid diaphragms can grossly misrepresent their actual behaviour'.

Yet with all these possible approaches it still appears that one question remains unanswered, that being the comparative merits of these various solution processes. When searching for a program the designer must choose one to suit his requirements, but will he know what the various programs will provide and what is more important, will the one he chooses be the most efficient? Thus it appears that any information on the comparative merits of program systems for elastic analysis would be beneficial.

1.2 The Scope of Proposed Work.

With the foregoing background of research in the field of three-dimensional structures it was decided to develop a computer program to analyse a special, though by no means rare, kind of structure. It was hoped that techniques utilised in the work on Tee-beam analysis[13] would prove fruitful in reducing the problem to a manageable size.

The specification of the kind of structure to be considered was not expected to be restrictive since the nature of the specialised structure was to be orthogonal, a type which occurs frequently in today's framed buildings. The structure could consist of column and beam elements interconnected by floor slabs. The construction material must be borne in mind since it may also influence the nature of the program.

It is necessary to try to avoid the assumption that the floor slabs act as rigid diaphragms since this can lead to

erroneous results[12]. Thus it was essential to establish suitable mathematical representation of the floor elements and an investigation into this problem was scheduled, bearing in mind the need for the ease with which element formulation would allow versatility in the floor layout.

The main objective resolved itself into a full three-dimensional elastic analysis of orthogonal beam/column systems interconnected by elastic plate elements representing the floor deck. The developed program would utilise available mathematical and computational techniques in performing the analysis of multi-storey structures.

It was also decided to investigate the comparative merits of certain approaches to elastic structure analysis using programs which were available, these being a plane frame analysis, a skeletal space frame analysis and a 'complete structure' analysis. The results from this work may influence the choice of program type for a particular use, and would enable conclusions to be drawn as to how three-dimensional behaviour can be represented more accurately in a simpler analysis.

1.3 The Application of Programs for Elastic Analysis to Design.

When considering the problem of structural analysis it is important to realise the context for such an analysis. It is usually required in order to understand structural behaviour and to enable a 'safe' design against collapse or serviceability.

It could also be used to investigate the optimisation of a structural layout.

Analysis can be used for the overall structural behaviour in conceptual design or in determining element forces to facilitate element design. Therefore it is necessary to specify the objective in order to set the requirements of the computer program. The mass of results provided by large general purpose programs is not always applicable to certain stages of design and a plane frame analysis would provide the required information in a sufficiently accurate form.

The question of efficiency and economics must be considered when computer programs are to be used as a design tool. Unless the program can be demonstrated to be, in effect, more efficient and/or more economic than the methods it is replacing, its use cannot be said to be beneficial. The need for its operation to dove-tail into the production side of design work is of particular importance when it comes to user orientated programming. Yet this cannot be an overriding factor such that the programs must be adapted to cater for general office practice. However, incorporation of a program system should not cause undue interference with the efficient running of the design process.

1.4 The Present State of the Proposed Research.

This is a short resumé of the present condition of the work proposed in 1.2. The 'complete structure' program has been

implemented but only in a prototype form; however, successful use of the program has still been achieved.

Two other programs have been written or adapted, one analysing elastic plane frames and the other skeletal space frames, which were used together with the prototype program in the investigation of different elastic frame solutions.

The comparative frame testing was carried out with the three programs as stated above but the author would have wished to have extended the work in this field. Yet results for the tests completed have already shown interesting trends and have proved to be quite enlightening. The approach of considering a limited sub-frame concept in three-dimensions has not been successful but has lead to some encouraging results.

Despite a certain amount of progress being made, it is thought that there is still room for improvement and refinement to the main program and that further frame testing would eventually lead to beneficial results.

1.5 References.

1. R. K. Livesley - 'Matrix Methods of Structural Analysis' Pergamon Press, London (1964).
2. O. C. Zienkiewicz - 'The Finite Element Method in Engineering Science' McGraw Hill, London 1971.
3. J. G. A. Croll & A. C. Walker - 'The Finite Difference and Localised Ritz Methods' Int. Journal for Numerical Methods in Engineering Vol. 3 155-160 (1971).

4. J. G. A. Croll - 'Hermitian Methods for the Approximate Solution of Partial Differential Equations' J. Inst. Math. Applics. (1970) 6, 365-374.

5. F. M. Salonen - 'Rectangular plate bending element the use of which is equivalent to the use of the finite difference method'. Int. J. of Numerical Methods in Engineering. Vol. 1, 261-274 (1969).

6. R. W. Clough et al - 'Structural analysis of multi-storey buildings.' Proceedings of A.S.C.E. 1964, 90, ST3,19-34.

7. J. R. W. Weaver & M. F. Nelson - 'Three-dimensional analysis of tier buildings'. Journal Str. Div. A.S.C.E. ST6, 1966 (Dec) 385-404.

8. K. I. Majid & M. Williamson - 'Linear Analysis of Complete Structures by Computers'. Proceedings of I.C.E. Vol. 38 Oct. 1967, 247-266.

9. D. Bond - 'A Computer program for studying the design of reinforced concrete structures supported on columns'. Proceedings of I.C.E. Vol. 43, June 1969, 195-214.

10. M. C. Stamato & B. Stafford Smith - 'An approximate method for three-dimensional analysis of tall buildings'. Proceedings of I.C.E. Vol. 43, July 1969, 361-379.

11. O. C. Zienkiewicz, C. J. Parekh & B. Teply - 'Three-dimensional analysis of buildings composed of floor and wall panels'. Proceedings of I.C.E. Vol 49, July 1971, 319-332.

12. K. I. Majid & P. C. L. Croxton - 'Wind analysis of complete building structures by influence coefficients'. Proceedings of I.C.E. Vol 47, Oct 1970, 169-184.

13. See section 2.

2. THE DETERMINATION OF POINTS OF CONTRAFLEXURE ALONG TEE-BEAMS.

2.1 Introduction.

In current computer programs for the elastic analysis of plane frames by the stiffness method, the stiffness of each member of the frame is constant, and where it is desired to include a member of non-uniform cross-section it is necessary to introduce new joints at the positions of the change in properties.

In analysing concrete plane frames, in a monolithic structure, the beams can be assumed to act as Tee-beams. Here, there is an increase in the beam's stiffness produced by the increase in cross-sectional area of the member. However, since Tee-beam action occurs only in the compression part of the member, the analyst has to choose whether to make the whole, or part of, the beam a Tee-section.

In the first case no new joints are needed but it may be that a Tee-beam has been assumed where a hogging moment exists, i.e. a compression flange has been assumed where the beam is in tension. The usual form of the bending moment along the beam does, in fact, produce two such regions at each end of the beam. Thus the second choice appears more realistic but leads to the problem of where the change from a hogging to a sagging moment occurs.

The changeover points are known as the points of contraflexure and are not fixed but, in a fully rigid rectangular structure, depend on the system of loading and restraint on the beam.

In practice it appears that these points are assumed to be at fixed positions on the beam, varying between 0.1 and 0.15 of the length of the beam from its end. It seems that these values have been set by experience and are accepted in codes of practice. However, as there has been little work done in the determination of accurate positions for these points, considering Tee-beam action, a computer program was written which analyses plane frames using Tee-beams and which calculates the locations of the points of contraflexure. These calculated locations are then used to re-construct the stiffness matrix for the Tee-beam, followed by a re-cycling of the analysis for iteration towards accurate location for the points of contraflexure.

The program provides an understanding of how a sophisticated elastic analysis program is implemented. It enables an investigation into the use of a condensation technique, demonstrating its advantages but also illustrating its drawbacks. The Tee-beam analysis allows the study of the behaviour of a segmented beam, with particular reference to its effect on the bending moment distribution, to be undertaken.

The matrix techniques used are well documented and for further references the reader should refer to works by Livesley[1], Morice[2], Jenkins[3] or Rubinstein[4] and many others too numerous to mention.

2.2 Tee-beam stiffness matrix.

The stiffness matrix of a Tee-beam comprises three element stiffnesses, where the beam is assumed to be composed of three elements or segments. The two end segments are assumed to be acting under a hogging moment and therefore are taken to be rectangular in cross-section, whereas the middle segment is in compression and thus is assumed to have a Tee section. A stiffness matrix is constructed for each segment and the total stiffness matrix of the beam is obtained by combining the segment matrices.

A compatible matrix for a Tee-beam, of the same form as that for an ordinary member, is obtained by the removal of the unwanted displacements. This is accomplished by suitable adjustments to the total stiffness matrix to prepare it for a Gaussian elimination procedure[5]. This procedure is stopped as soon as the unwanted displacements have been removed. The 'condensed stiffness matrix' for the Tee-beam is now contained in the total stiffness matrix after this elimination. This matrix will be the same size as the usual member stiffness matrix. The whole process is illustrated algebraically below:

Let r_1 be the vector of unwanted displacements, i.e. internal member displacements.

Let r_2 be the vector of retained displacements, i.e. member-end displacements.

Let R_1 and R_2 be the corresponding load vectors.

K is the stiffness matrix so that

$$K. \begin{vmatrix} r_1 \\ \\ r_2 \end{vmatrix} = \begin{vmatrix} R_1 \\ \\ R_2 \end{vmatrix} \qquad (2.1)$$

where $K = \begin{vmatrix} k_1 & k_2 \\ k_2^T & k_3 \end{vmatrix}$ (2.2)

therefore $k_1.r_1 + k_2.r_2 = R_1$

$k_2^T.r_1 + k_3.r_2 = R_2$ and eliminating r_1

$$(k_3 - k_2^T.k_1^{-1}.k_2).r_2 = R_2 - k_2^T.k_1^{-1}.R_1 \qquad (2.3)$$

which is of the usual form $\underline{K}.r = \underline{R}$

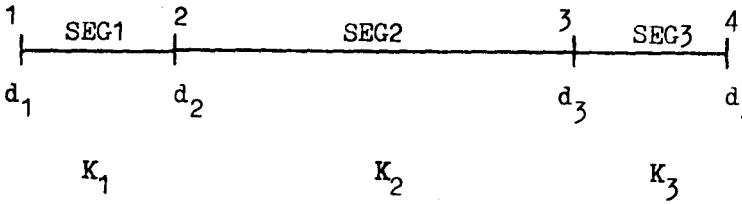
where the condensed matrix $\underline{K} = (k_3 - k_2^T.k_1^{-1}.k_2)$ (2.4)

and the condensed load vector $\underline{R} = R_2 - k_2^T.k_1^{-1}.R_1$ (2.5)

It should be noticed that this condensed matrix is load dependent which separates it from the usual load independent member stiffness matrices, since, when condensing a member matrix, it is necessary to condense the load vector associated with it.

How the condensation process is applied to the total stiffness matrix is not completely self-evident. The actual process is carried out by applying a Gaussian triangularisation routine to the total stiffness matrix of the Tee-beam after it has been re-arranged so that the unwanted and retained displacements are in the vectors r_1 and r_2 respectively as shown in equation (2.1). Fig 2.1 shows the Tee-beam segment matrices and their subsequent 'addition' to form the total stiffness matrix.

Fig. 2.1 Segmented Tee-beam.



where K_i (i=1,2,3) is of the form $\begin{vmatrix} k_{i1} & k_{i2} \\ k_{i3} & k_{i4} \end{vmatrix}$ (2.6)

Thus the total stiffness matrix of the Tee-beam is $\underline{K}$ where

$$\underline{K}.D = R \qquad (2.7)$$

where D is the total displacement vector $\begin{vmatrix} d_1 & d_2 & d_3 & d_4 \end{vmatrix}^T$ and R is the total load vector.

The matrix $\underline{K}$ in equation (2.7) is of the form

$$\begin{vmatrix} K_1 & & \\ & K_2 & \\ & & K_3 \end{vmatrix} \begin{vmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{vmatrix} = \begin{vmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \end{vmatrix} \qquad (2.8)$$

Now the actual end displacements of the Tee-beam are contained in vectors d_1 and d_4, leaving d_2 and d_3 as the unwanted displacements represented by r_1 in equation (2.1).

It is possible to expand equation (2.8) using the individual segment matrices K_i as expressed in equation (2.6) for the total stiffness matrix $\underline{K}$. Then after rearranging this matrix to obtain the form of equations shown in (2.1) and (2.2) equation (2.8) becomes

$$\left|\begin{array}{cc|cc} k_{14}+k_{21} & k_{22} & k_{13} & 0 \\ k_{23} & k_{24}+k_{31} & 0 & k_{32} \\ \hline k_{12} & 0 & k_{11} & 0 \\ 0 & k_{33} & 0 & k_{34} \end{array}\right| \left|\begin{array}{c} d_2 \\ d_3 \\ \hline d_1 \\ d_4 \end{array}\right| = \left|\begin{array}{c} R_2 \\ R_3 \\ \hline R_1 \\ R_4 \end{array}\right| \qquad (2.9)$$

This is now in the equivalent form of $\left|\begin{array}{cc} k_1 & k_2 \\ k_2^T & k_3 \end{array}\right| \left|\begin{array}{c} r_1 \\ r_2 \end{array}\right| = \left|\begin{array}{c} R_1 \\ R_2 \end{array}\right|$

since $k_{i2}^T = k_{i3}$.

Therefore it is now possible to eliminate the unwanted displacements using the Gaussian process. A flowchart for the procedure is given in Fig. 2.2, where it is sufficient to note that if the process is operated to a required depth of the stiffness matrix (in equation (2.9)) a condensed matrix and load vector will be obtained.

2.3 Points of contraflexure.

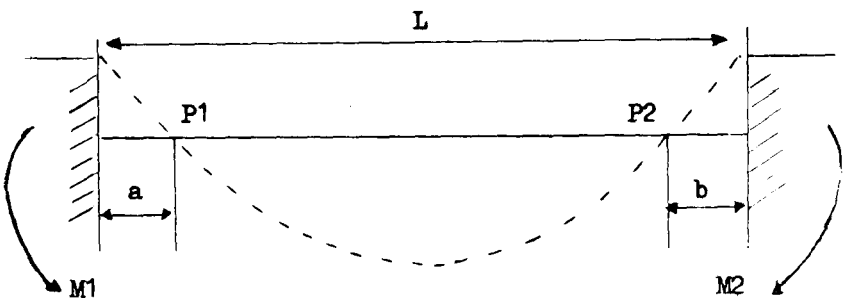


Fig. 2.3 General form of Bending Moment Distribution along a Tee-beam.

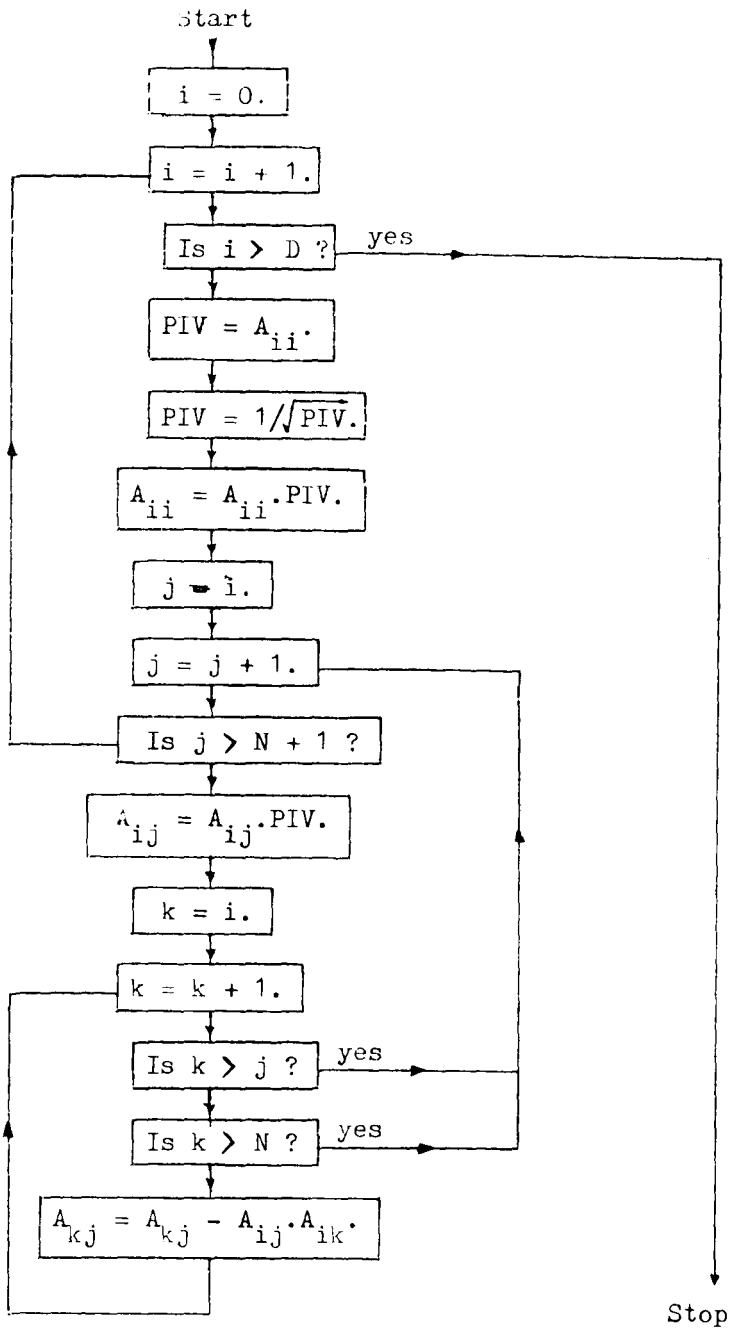


Fig. 2.2 Flowchart diagram for matrix condensation process.

If the total stiffness matrix is $\underline{K}$ with load vector R then the matrix A is as follows:- $A = \left| \underline{K} \;\middle|\; R \right|$

i.e. the last column of A holds the load vector.

The size of A is N N+1.

The required depth of condensation is D.

Fig. 2.3 illustrates the usual form of the bending moment which exists along a typical Tee-beam. It is required of the program to determine the points of contraflexure on those beams being used as Tee-beams. Since these positions are the locations of zero moment along the beam, the full bending moment distribution along the beam is required. In Fig. 2.3 the points of contraflexure are indicated by P1 and P2.

The joint displacements and forces are obtained from the stiffness matrix analysis of the plane frame. Therefore the restraining moments, labelled M1 and M2 in Fig. 2.3, are obtained from this nodal analysis, and the total moment distribution is the sum of the restraining moment and free bending moment distributions. Thus the only requirement is the so called 'simply supported' moment distribution. This is not readily obtainable from the analysis and extra procedures are necessary to store member loads in readiness for its calculation. Using these loads and member properties the simply supported moment distribution for any member is obtained via the well known formulae for point, uniformly distributed and moment loads, which are built into the program.

Having obtained the total distribution it is necessary to locate the points of contraflexure. Specific values for the moment are stored at N+1 points at intervals of 1/N along the beam. A search is then made to locate a change of sign between M_i and M_{i+1} (the moment values at the i^{th} and $i+1^{th}$ locations). Where a change of sign is obtained, it is known that a point of contraflexure exists in the interval (i, i+1). The exact location is not easily obtained without recourse to successive shortening of the interval. However, an approximate position can be obtained

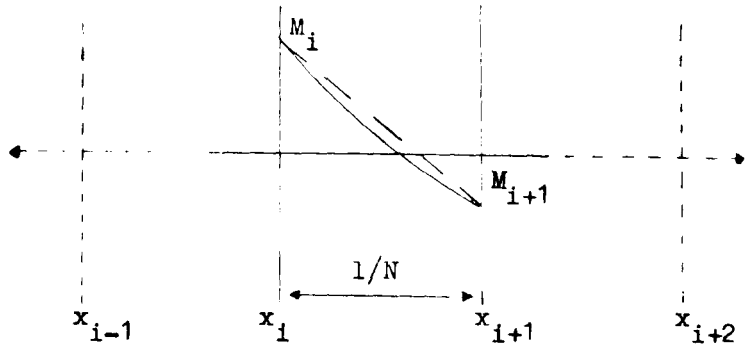


Fig. 2.4 Linear interpolation, after interval search of bending moment for change in sign.

——— Exact moment curve.

- - - - Linear approximation to moment curve.

l Beam length.

x_i i^{th} x coordinate along beam.

by a simple linear interpolation between the interval, as illustrated in Fig. 2.4. The position obtained is an approximation but has proved sufficiently accurate for general analysis purposes. Due to the usual shape of the bending moment curve, the slope between an interval is fairly shallow, tending to a straight line, and therefore a linear interpolation is a reasonable first approximation. The part of the program concerned with this section will find the number of points of contraflexure together with their approximate locations and would expect zero, one or two points per beam.

2.4 Program Implementation.

The techniques previously outlined for the Tee-beam matrix

assembly and for the determination of the points of contraflexure are both introduced into a simple elastic plane frame program. The one chosen for this purpose was supplied by the Structural Computation Unit at the University of Warwick. The program was a stiffness matrix analysis but with a highly sophisticated user-orientated input/output system.

An iteration process is set up to utilise calculated locations for the points of contraflexure as a better assessment of Tee-beam stiffness. However, initial positions are required and they are set at a tenth of the length from each end of the beam. The iteration continues until successive positions of the points of contraflexure have converged to a desired accuracy. The convergence limit of 0.05 ins (0.1 mms) has been built into the program and has proved suitable for general use.

The whole procedure is illustrated by the flowchart given in Fig. 2.5, where the new sections are labelled with an asterisk. An operation manual for potential users has been issued by the Structural Computation Unit (Manual No. 7 - referring to Manuals Nos. 1 and 5).

Basically there are only two major new additions to the main body of the original program. The first handles the Tee-beam members' stiffness assembly, utilising some of the procedures listed in Table 2.1. The second is a more independent segment which handles the solution for the points of contraflexure. This segment has been given the name INFLEC in the program and it also utilises certain procedures given in Table 2.1.

TABLE 2.1 Description of new procedures in Tee-beam program.

NAME	INPUT	DESCRIPTION
TB	section dimensions	calculates Tee-beam area a inertia.
AVOID	--	constructs structure load v from stored member loads.
STORELOAD	member loading	stores member loads for fut use.
IFEF	member loading	calculates the fixed end ef due to member loading for a segment of a Tee-beam for u the total matrix load vecto
CONDES	Tee-beam matrix and load vector	condensation routine.
POSMEM	segment member matrix	positions individual Tee-be segment matrices into the t matrix ready for condensati
PICKUP	condensed matrix and load vector	stores the condensed Tee-be matrix and vector ready for solution of member forces.
SOLVTB	--	calculates Tee-beam member end forces.

TABLE 2.1 continued.

NAME	INPUT	DESCRIPTION
MOMENT	member loading and fixed end forces	forms the bending moment distribution along a Tee-b
ROOTS	moment distribution, member length and intervals per beam	finds the number and posit of zero moment along a Tee-
REINT	old and new segment lengths	resets the segment lengths Tee-beams from the latest known positions for the points of contraflexure.
PUNOUT	latest segment lengths	output procedure for final Tee-beam segment lengths.
CONV	old and new points of contraflexure	checks on the convergence o successive positions of the points of contraflexure.

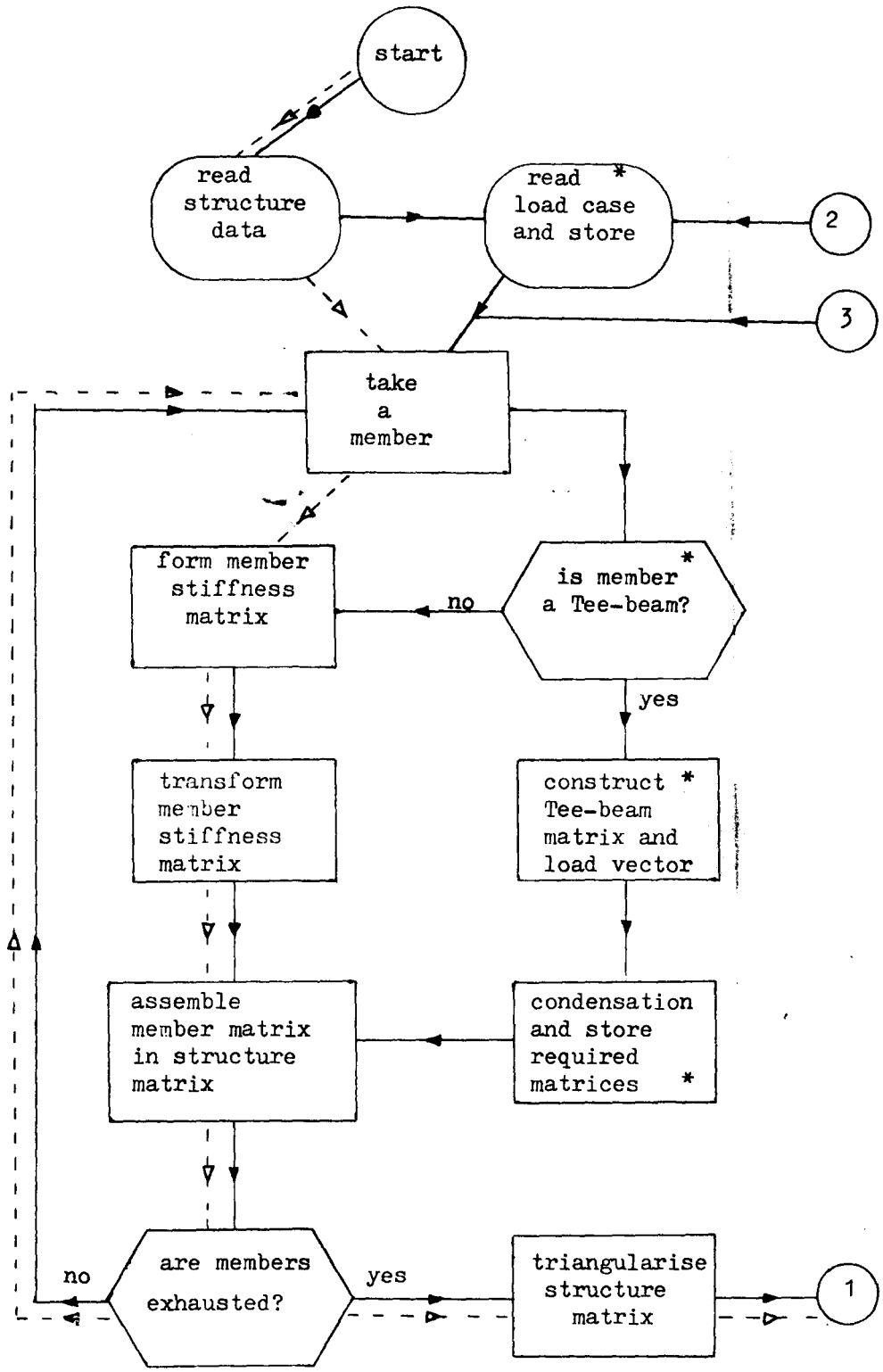


Fig. 2.5 Flowchart for Tee-beam program.

——▶—— Tee-beam program route.

— -▷- — Original program route. * New procedures.

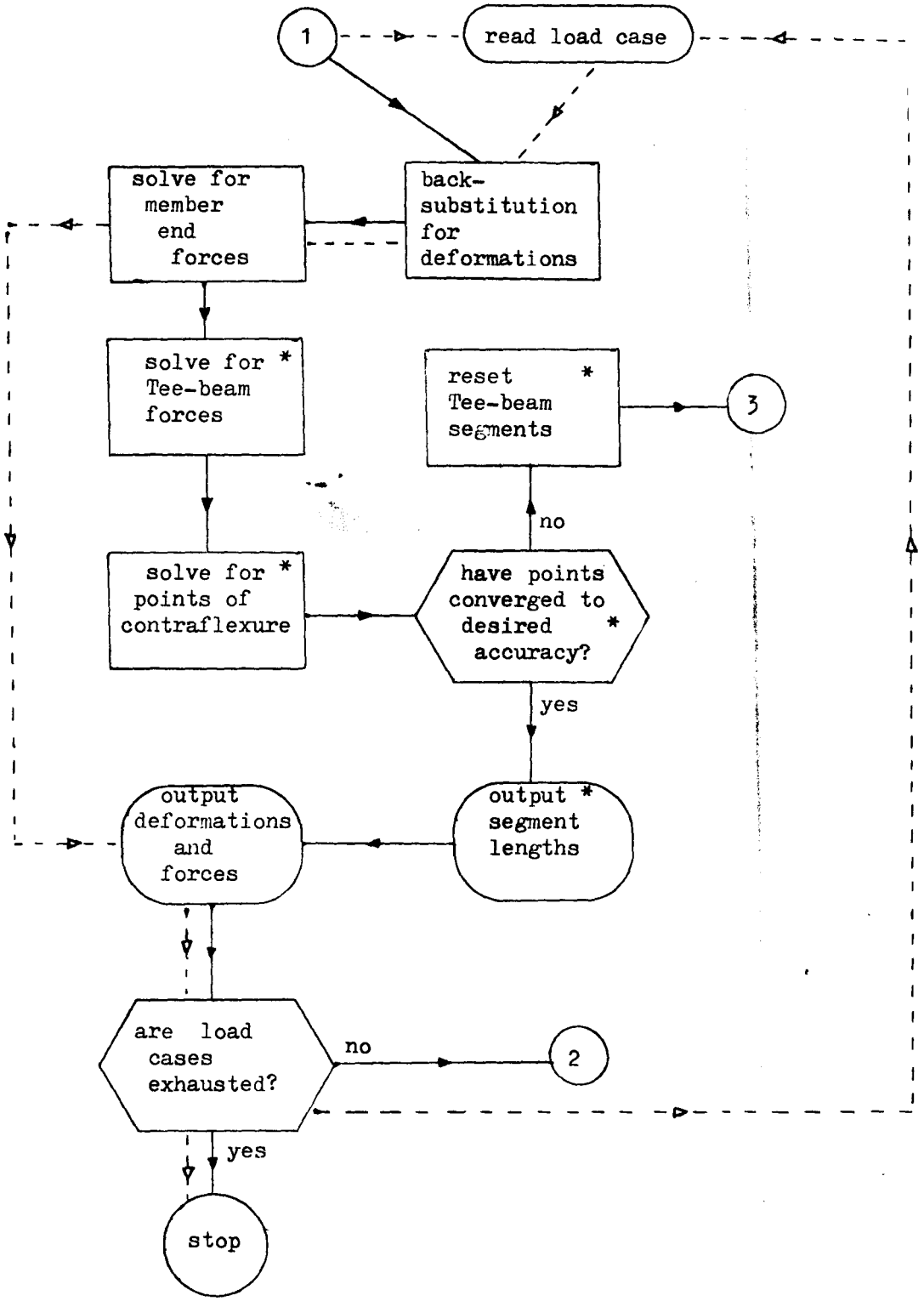


Fig. 2.5 continued. Flowchart for Tee-beam program.

It is evident from the flowchart of Fig. 2.5 that a slight re-structuring of the program was necessary. This was in the positioning of the input load data. Previously the structure stiffness matrix was composed of purely load independent member matrices, but now, with the inclusion of Tee-beams whose condensed matrix is load dependent, it was necessary to re-position the load input section. This is illustrated by the flowchart which gives both the original and new positions.

The program, already having a user-orientated input system, will handle various types of member loading. This can comprise of any combination of either line, point or moment loads applied at specified locations on a member. Therefore it was required to allow the Tee-beam members to also be loaded in this way.

An extra facility added to the Tee-beam program was one to enable the self-weight of a concrete structure to be included in the frame analysis. This was done internally by specifying a concrete density - 150 lb/cu.ft (24 KN/M^3) - and using the gross cross-section and length to calculate member self-weight. At the input stage it is possible to select those members whose self-weight is to be considered in the analysis.

The information for the points of contraflexure is given by displaying the segment lengths of the Tee-beam, assuming that usually there are three segments. Also the type of cross-section for each segment is given. The number of beam

intervals is set at 100.

As an example, for a Tee-beam member of length 120ins., a typical output could be as follows:-

T-BEAM SEGMENTS

LENGTH IN INCHES

MEMBER	SEG1	SEG2	SEG3
X-SECTION	RECT	TEE	RECT
2 - 3	7.4	105.3	7.4

Fig. 2.6 Example points of contraflexure output.

If only one point exists then the relevant segment will be put to zero length, and in the case where the moment is purely hogging - i.e. no points exist - the middle segment will revert to 'RECT' in cross-section. Also output is the number of iterations taken for all inflection points to have converged to the stated accuracy.

The program is written in 4100 Algol and was developed on an I.C.L. 4130 machine, housed at the University of Warwick. It has been stored in object code on disc ready for use in a batch system process.

It was tested by comparing results with those obtained from 'ELAN5', the plane frame analysis program from which the Tee-beam one was developed. The test consisted of first using 'Tee-beam' to establish the locations of the points

of contraflexure. Then, a similar frame was simulated for use with the 'ELAN5' program. This was done by introducing new nodes at the points of contraflexure so that different properties could be given for each segment of the beam corresponding to rectangular or tee section as required. The 'ELAN5' output gave the moment values at the positions of contraflexure and these were expected to be zero if the locations were exact. The values obtained were indeed very small and it was established that the moments given at determined points of contraflexure were less than 1% of the smallest end moment of each beam. The results obtained from this test were believed to be satisfactory and well within any working limits required. The test frame used here is the same as that given in the following results section - Example Frame 1.

2.5 Example Frame Illustrating the Use of the Tee-beam Program.

The example chosen is the simple rectangular framed structure shown below in Fig. 2.7. The actual input and output is given, together with the structure properties. For a specification of the use of the program the reader is referred to Manual No. 7 issued by the Structural Computation Unit at the University of Warwick.

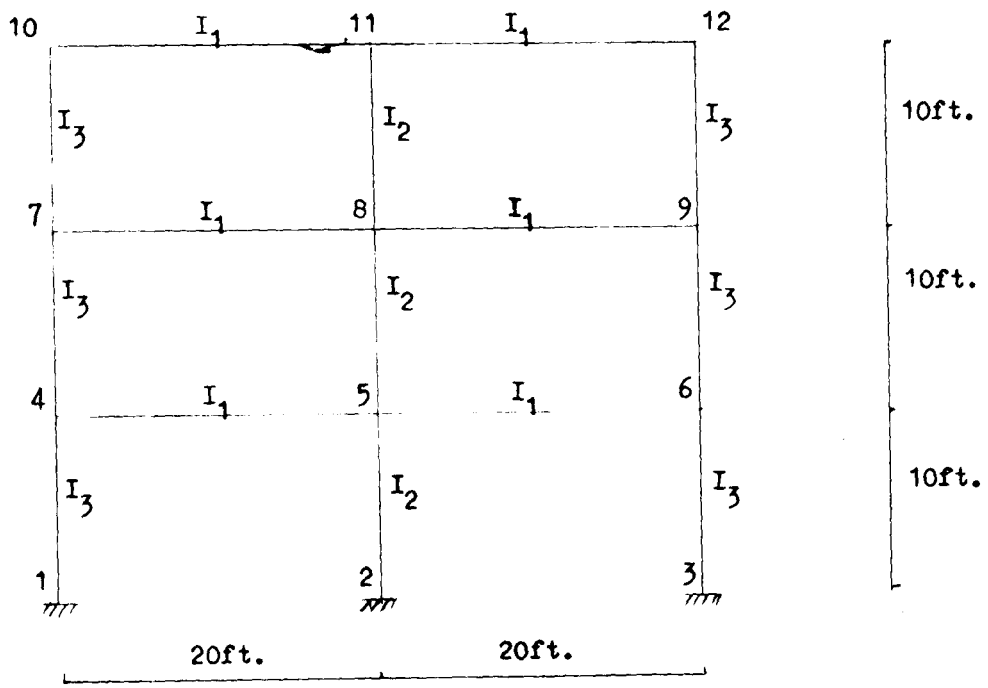


Fig. 2.7 Example Frame 1.

Member Properties: In Inches		Breadth	Depth	Flange	Floor Depth
	I_1	15	18	75	5
	I_2	12	12		
	I_3	9	12		

Young's Modulus - 2×10^6 lbs per sq.in.

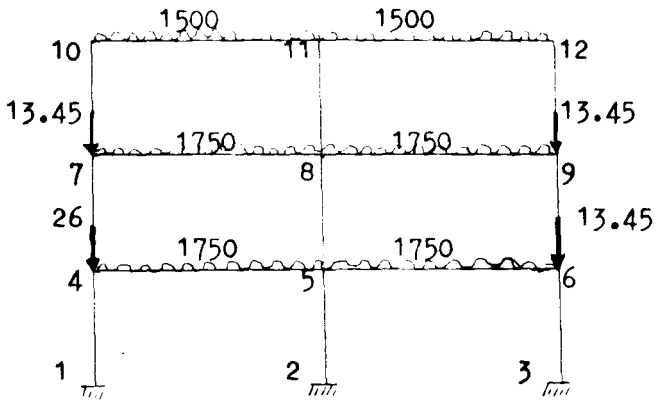


Fig. 2.8 Loading Case 1.

Dead Load.

Super Load.

Self-weight of members.

Units: Line loads in lb/ft run, point loads in Kips.

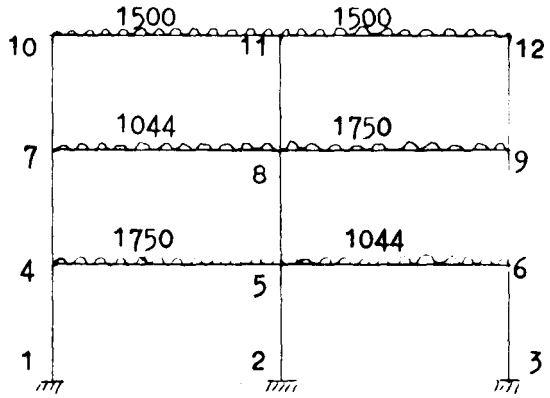


Fig. 2.9 Loading Case 2.

Dead Load.

Super Load on alternate spans.

Self-weight of structure.

Units: Line loads in lb/ft run.

The two loading cases considered are shown in Figs. 2.8 and 2.9.

Results for the example.

The full computer output is now given together with the moment distribution on the frame for load case 2 in Fig. 2.10.

```
[EXAMPLE FRAME]
9;12;15;2;6;
IMPERIAL CONCRETE
1;XYR 2;XYR 3;XYR
*
1/4/7/10;3/6/9/12(101)
2/5/8/11(102)
4/5/6;7/8/9;10/11/12(103)
101)9;12;
102)12;12;
103)15;18;75;5;
*
0;20;40;0;20;40;0;20;40;0;20;40;
0(3;11(3;21(3;31(3;
*

[DEAD +SUPER LOAD]
L-1750 V 4/5;5/6;7/8;8/9;
L-1500 V 10/11;11/12;
P -26 V 4;
P -13.45 V 6;7;9;
L -131 S 1/4;4/7;7/10;3/6;6/9;9/12;
L -150 S 2/5;5/8;8/11;
*
```

ALL CONVERGED

ITERATIONS EQUAL 3

T-BEAM SEGMENTS

LENGTHS IN INCHES

MEMBER	SEG1	SEG2	SEG3
X-SECTION	RECT	TEE	RECT
4- 5	16.9	176.9	46.2
5- 6	46.2	176.8	17.0
7- 8	20.3	174.4	45.3
8- 9	45.3	174.4	20.3
10- 11	12.5	181.7	45.7
11- 12	45.7	181.7	12.5

CASE 1

DEFLECTIONS IN INCHES ROTATIONS IN RADIANS X 100

JT.	X-DIRN	Y-DIRN	ROTN
1	0.0000	0.0000	0.0000
2	0.0000	0.0000	0.0000
3	0.0000	0.0000	0.0000
4	-0.0014	-0.0530	-0.1155
5	-0.0010	-0.0531	0.0016
6	-0.0006	-0.0453	0.1186
7	-0.0029	-0.0775	-0.1056
8	-0.0029	-0.0844	0.0016
9	-0.0029	-0.0698	0.1088
10	-0.0042	-0.0850	-0.1216
11	-0.0048	-0.0989	0.0016
12	-0.0055	-0.0773	0.1248

FORCES KIPS MOMENTS KIPS FT

MEMBER	AXL1	SHR1	MOM1	AXL2	SHR2	MOM2
1- 4	87.47	-1.05	-3.88	-86.03	1.05	-7.66
4- 7	44.67	-2.42	-12.26	-43.36	2.42	-11.90
7- 10	14.23	-2.48	-12.09	-12.92	2.48	-12.67
3- 6	74.93	1.05	3.83	-73.49	-1.05	7.71
6- 9	44.67	2.42	12.25	-43.36	-2.42	11.90
9- 12	14.23	2.48	12.09	-12.92	-2.48	12.67
2- 5	116.72	0.00	-0.03	-115.07	-0.00	0.03
5- 8	75.80	0.00	-0.00	-74.30	-0.00	0.00
8- 11	35.65	-0.00	-0.00	-34.15	0.00	0.00
4- 5	-1.37	15.36	19.92	1.37	19.64	-62.66
5- 6	-1.37	19.63	62.62	1.37	15.37	-19.97
7- 8	-0.06	15.67	23.99	0.06	19.33	-60.50
8- 9	-0.06	19.33	60.50	0.06	15.67	-23.99
10- 11	2.48	12.92	12.67	-2.48	17.08	-54.18
11- 12	2.48	17.08	54.18	-2.48	12.92	-12.67

[DEAD + SUPER ON ALT. SPANS]

L -1750 V 4/5;8/9;

L -1044 V 5/6;7/8;

L -1500 V 10/11;11/12;

L -150 S 2/5;5/8;8/11;

L -131 S 1/4;4/7;7/10;3/6;6/9;9/12;

*

ALL CONVERGED

ITERATIONS EQUAL 3

T-BEAM SEGMENTS

LENGTHS IN INCHES

MEMBER	SEG1	SEG2	SEG3
X-SECTION	RECT	TEE	RECT
4- 5	17.0	185.1	37.9
5- 6	55.8	164.5	19.6
7- 8	26.0	160.1	53.9
8- 9	36.4	183.1	20.6
10- 11	11.8	185.0	43.1
11- 12	45.2	181.3	13.5

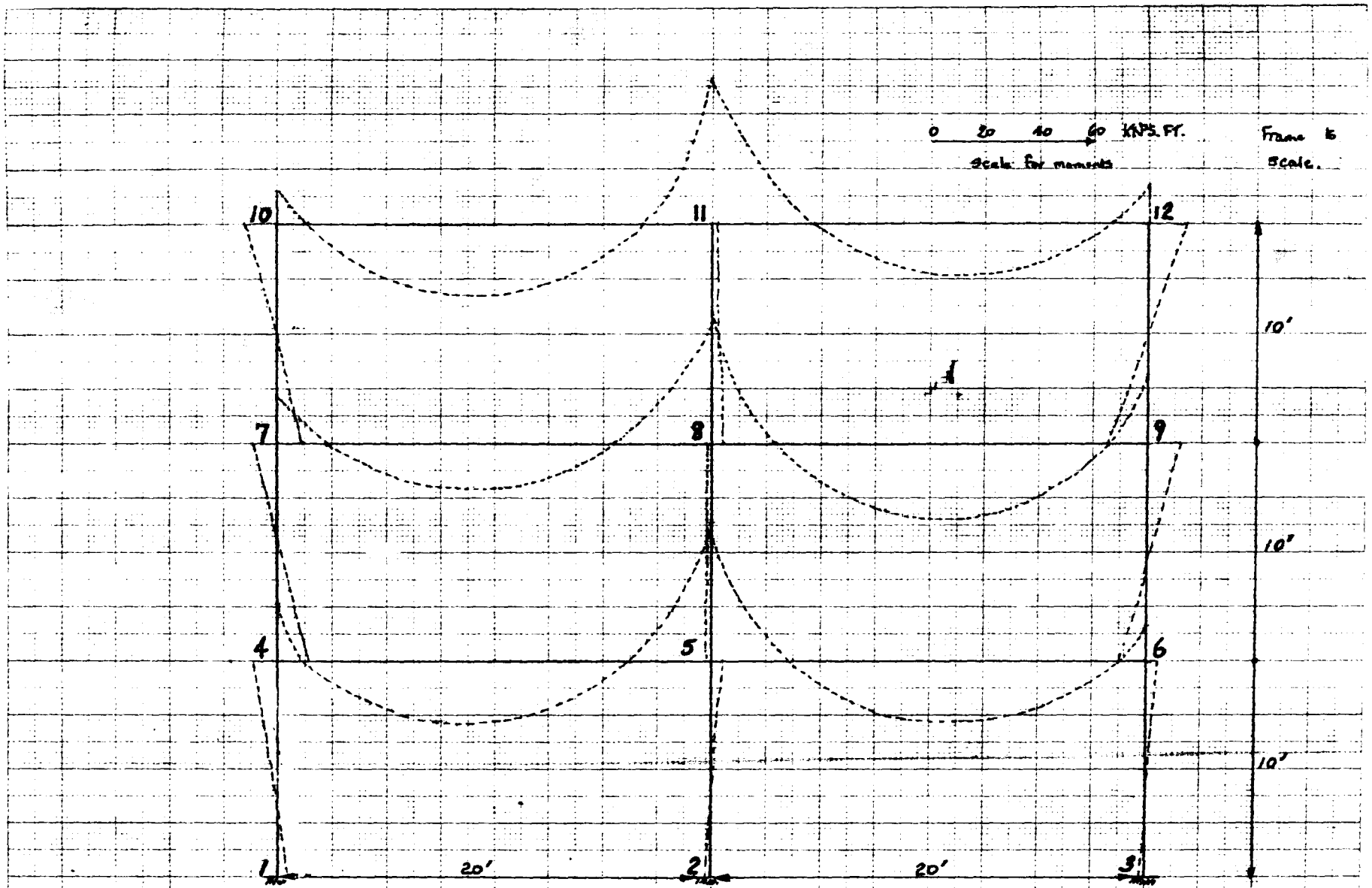


Fig 2.10 Moment distribution for frame of example – load case 2.

CASE 2

DEFLECTIONS IN INCHES ROTATIONS IN RADIANS X 100

JT.	X-DIRN	Y-DIRN	ROTN
1	0.0000	0.0000	0.0000
2	0.0000	0.0000	0.0000
3	0.0000	0.0000	0.0000
4	0.0039	-0.0254	-0.1412
5	0.0042	-0.0454	0.0462
6	0.0046	-0.0254	0.0582
7	0.0045	-0.0389	-0.0506
8	0.0045	-0.0731	-0.0466
9	0.0047	-0.0428	0.1318
10	0.0025	-0.0465	-0.1405
11	0.0020	-0.0876	0.0056
12	0.0012	-0.0504	0.1289

FORCES KIPS MOMENTS KIPS FT

MEMBER	AXL1	SHR1	MOM1	AXL2	SHR2	MOM2
1- 4	42.31	-1.21	-4.33	-40.87	1.21	-8.95
4- 7	24.89	-2.06	-11.93	-23.58	2.06	-8.67
7- 10	14.35	-2.10	-8.88	-13.04	2.10	-12.12
3- 6	42.29	0.58	2.25	-40.85	-0.58	4.15
6- 9	31.99	2.06	8.95	-30.68	-2.06	11.60
9- 12	14.33	2.75	13.82	-13.02	-2.75	13.71
2- 5	99.93	0.63	2.43	-98.28	-0.63	4.45
5- 8	67.24	0.00	2.25	-65.74	-0.00	-2.20
8- 11	35.43	-0.65	-4.51	-33.93	0.65	-2.01
4- 5	-0.85	15.98	20.88	0.85	19.02	-51.30
5- 6	-1.47	12.02	44.60	1.47	8.86	-13.10
7- 8	-0.04	9.23	17.55	0.04	11.65	-41.81
8- 9	-0.70	18.66	48.53	0.70	16.34	-25.42
10- 11	2.10	13.04	12.12	-2.10	16.96	-51.25
11- 12	2.75	16.98	53.26	-2.75	13.02	-13.71

Storage Used: 25k. Time: 57 seconds.

2.6 Some further test results using Tee-beam program.

A selection of some of the results obtained by use of the Tee-beam program in an investigation into Tee-beam behavior are presented. However, to limit the size of this section a full report will not be given.

1. Comparison of the positions of points of contraflexure and end moments of beams in the given example - Fig. 2.7 Frame 1 - using: (a) Tee-section beams and

(b) Rectangular section beams.

In the results the three segment lengths are given as a ratio of the total span length and the end moments are given in Kips.ft.

TABLE 2.2

Results for Load Case 1.

(a) Tee- section.					
MEMBER	SEGMENTS			END MOMENTS	
	TENSION	COMPRESSION	TENSION	M1	M2
4 - 5	0.07	0.74	0.19	19.92	-62.66
5 - 6	0.19	0.74	0.07	62.62	-19.97
7 - 8	0.08	0.73	0.19	23.99	-60.50
8 - 9	0.19	0.73	0.08	60.50	-23.99
10- 11	0.05	0.76	0.19	12.67	-54.18
11- 12	0.19	0.76	0.05	54.18	-12.67
(b) Rectangular Section.					
4 - 5	0.10	0.67	0.23	26.79	-73.80
5 - 6	0.23	0.67	0.10	73.76	-26.84
7 - 8	0.12	0.65	0.23	31.76	-70.84
8 - 9	0.23	0.65	0.12	70.83	-31.76
10- 11	0.08	0.69	0.23	17.36	-65.07
11- 12	0.23	0.69	0.08	65.07	-17.36

Fig 2.11 Variation of Tee-beam end moments with flange width for various floor depths of a fixed size rectangular section.

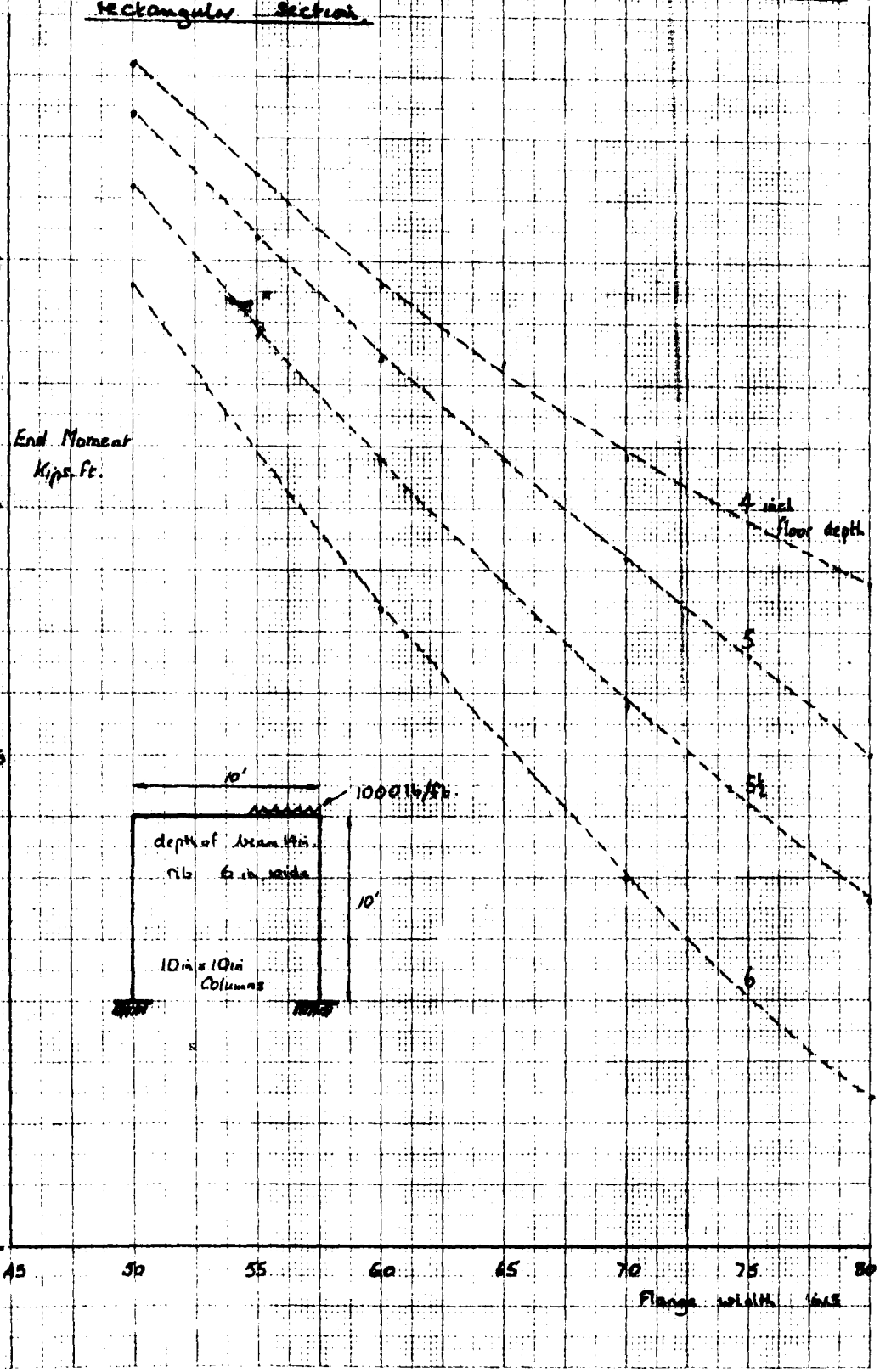


TABLE 2.3

Results for Load Case 2.

(a) Tee-section.

MEMBER	SEGMENTS			END MOMENTS	
	TENSION	COMPRESSION	TENSION	M1	M2
4 - 5	0.07	0.77	0.16	20.88	-51.30
5 - 6	0.23	0.69	0.08	44.60	-13.10
7 - 8	0.11	0.67	0.22	17.55	-41.81
8 - 9	0.15	0.77	0.08	48.53	-25.42
10 -11	0.05	0.77	0.18	12.12	-51.25
11 -12	0.19	0.75	0.06	53.26	-13.71
(b) Rectangular section.					
4 - 5	0.10	0.70	0.20	28.12	-62.15
5 - 6	0.28	0.61	0.11	52.53	-16.60
7 - 8	0.14	0.58	0.28	21.93	-49.05
8 - 9	0.19	0.69	0.12	58.65	-33.55
10 -11	0.07	0.71	0.22	16.45	-62.35
11 -12	0.24	0.68	0.08	65.02	-18.35

2. A single example of the variation of Tee-beam end moments with respect to flange width for four different floor thicknesses i.e. depth of the flange is presented. These results are shown graphically in Fig. 2.11. Only one load case is considered for the example.

2.7. Comments on the example of Fig. 2.7. and test results.

It can be seen from the results that the points of inflexion for the example do not distribute themselves evenly along the beam, tending to have larger tension zones over continuous supports. The difference is quite substantial for this regular

frame with uniform dead and live loads. Also the iterations taken for the analysis is given as three, but it must be borne in mind that this is not three times the total run time and therefore would not necessarily increase the run time costs by three.

The output of segment lengths enables the positions of curtailment of tensile reinforcement for span and support to be established. If it is desired to find the inflection points for rectangular beam systems, it is only necessary to use Tee-beams with a zero floor depth. This is how results given in Section 2.6 were obtained.

From the results given in TABLES 2.2 and 2.3 the size of the compression zone for the Tee-beam case ranges from 0.67 to 0.77 of the span. This agrees reasonably well with the values set in the New Unified Code[6] where, in Section 3.3.1.2, 'for a continuous beam the distance between the points of zero moment may be taken as 0.7 times the effective span'. A comparison of results for Tee-beam and rectangular beams shows the increase in the compression zone by use of a Tee-beam and also the reduction in member-end moments, due to the moment attraction of the stiffer Tee-section.

This moment attraction is also shown in Fig. 2.11, where the variation of end moment with flange width is illustrated. However, only the single case, that of a simple frame with uniform loading and a fixed size rib section, is given. A family of curves is shown for varying flange depths. Since the results shown are in isolation it is impossible to draw any quantitative

results. Further investigations have been undertaken, however, results of these will not be presented in this short report.

2.8 Summary and Conclusions.

The Tee-beam program is in a completed user state and has performed highly efficiently, succeeding in using the condensation technique to advantage. The iteration process and convergence limit have proved adequate for various test frames with irregular loading. However, an improvement to the program would appear possible whereby the bending moment envelope along the beam could be added to the output.

The test results given illustrate the field of investigation which has been undertaken. However, as stated earlier, results of these will not be presented here. Yet the line of research appears beneficial and further work in this area would seem appropriate.

2.9 References.

1. R. K. Livesley - 'Matrix Methods of Structural Analysis' Pergamon Press, London 1964.
2. P. B. Morice - 'Linear Structural Analysis' Thames & Hudson, 1959.
3. W. M. Jenkins - 'Matrix and Digital Computer Methods in Structural Analysis' McGraw Hill, London 1969.

4. M. F. Rubinstein - 'Matrix Computer Analysis of Structures' Prentice-Hall 1966.

5. G. Bull - 'Computational Methods and Algol' chp. 8, p. 174-192, Harrap, London 1966.

6. Code of Practice for the Structural Use of Concrete. CP 110, Part 1, B.S.I., London 1972.

3. ORTHOGONAL SPACE FRAME ANALYSIS.

3.1 Introduction.

A program has been produced to analyse rigid orthogonal space frames, in which the frames are solved elastically by means of a stiffness matrix analysis. The program is not unique, there being many space frame programs in existence. However, this program has been developed only for the testing of other programs.

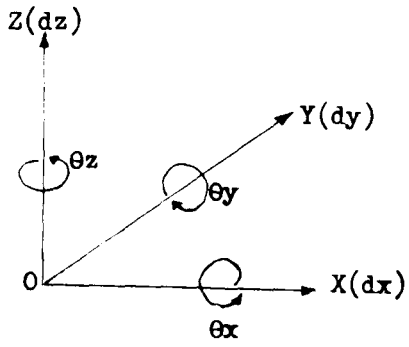
This is as follows:-

(a) in result testing of development work on more complex computer programs

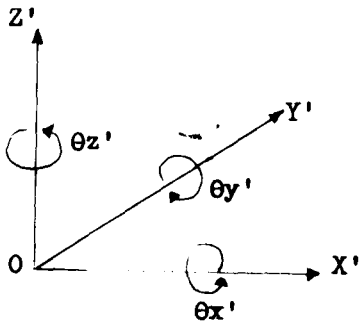
and (b) in comparative result testing of similar frames analysed by different programs.

Therefore this program has been restricted to rigid orthogonal skeletal space frames to facilitate ease of programming and development. Rigid connections are assumed throughout but it would be simple to convert the program for pinned connections. In a general analysis the transformation from member coordinates to structure coordinates becomes highly complex, especially if a further variable is introduced; that being the orientation of the principal axes to the longitudinal axis of the member. Thus to remove this complexity all members in a frame are assumed to be orthogonal in space and have their principal axes parallel to the member axes.

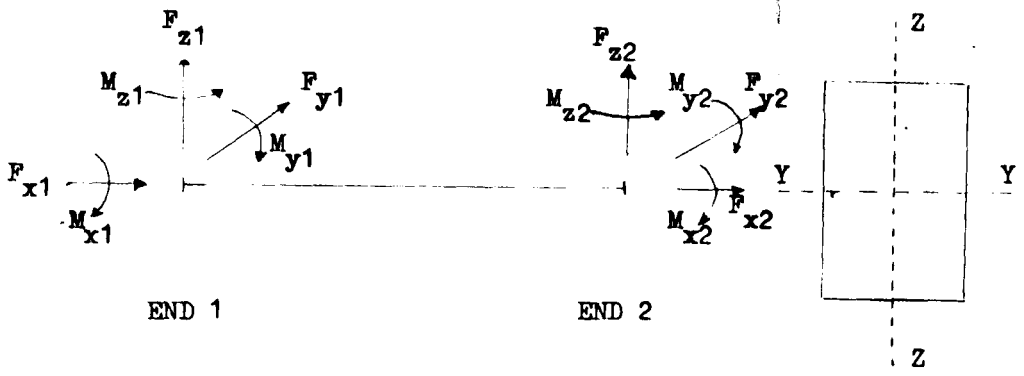
Therefore this program is by no means the most sophisticated



(a) Member coordinate axes and displacements.



(b) Structure coordinate axes.



(c) Member-end Forces. Member Section

Fig. 3.1. Coordinate Systems.

$$
\begin{vmatrix} M_{x1} \\ M_{y1} \\ M_{z1} \\ F_{x1} \\ F_{y1} \\ F_{z1} \\ M_{x2} \\ M_{y2} \\ M_{z2} \\ F_{x2} \\ F_{y2} \\ F_{z2} \end{vmatrix} =
\begin{vmatrix}
\frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 & \frac{-GJ}{L} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{4EI_y}{L} & 0 & 0 & 0 & \frac{-6EI_y}{L^2} & 0 & \frac{2EI_y}{L} & 0 & 0 & 0 & \frac{6EI_y}{L^2} \\
0 & 0 & \frac{4EI_z}{L} & 0 & \frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{2EI_z}{L} & 0 & \frac{-6EI_z}{L^2} & 0 \\
0 & 0 & 0 & \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & \frac{-EA}{L} & 0 & 0 \\
0 & 0 & \frac{6EI_z}{L^2} & 0 & \frac{12EI_z}{L^3} & 0 & 0 & 0 & \frac{6EI_z}{L^2} & 0 & \frac{-12EI_z}{L^3} & 0 \\
0 & \frac{-6EI_y}{L^2} & 0 & 0 & 0 & \frac{12EI_y}{L^3} & 0 & \frac{-6EI_y}{L^2} & 0 & 0 & 0 & \frac{-12EI_y}{L^3} \\
\frac{-GJ}{L} & 0 & 0 & 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{2EI_y}{L} & 0 & 0 & 0 & \frac{-6EI_y}{L^2} & 0 & \frac{4EI_y}{L} & 0 & 0 & 0 & \frac{6EI_y}{L^2} \\
0 & 0 & \frac{2EI_z}{L} & 0 & \frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{4EI_z}{L} & 0 & \frac{-6EI_z}{L^2} & 0 \\
0 & 0 & 0 & \frac{-EA}{L} & 0 & 0 & 0 & 0 & 0 & \frac{EA}{L} & 0 & 0 \\
0 & 0 & \frac{-6EI_z}{L^2} & 0 & \frac{-12EI_z}{L^3} & 0 & 0 & 0 & \frac{-6EI_z}{L^2} & 0 & \frac{12EI_z}{L^3} & 0 \\
0 & \frac{6EI_y}{L^2} & 0 & 0 & 0 & \frac{-12EI_y}{L^3} & 0 & \frac{6EI_y}{L^2} & 0 & 0 & 0 & \frac{12EI_y}{L^3}
\end{vmatrix}
\begin{vmatrix} \theta x_1 \\ \theta y_1 \\ \theta z_1 \\ dx_1 \\ dy_1 \\ dz_1 \\ \theta x_2 \\ \theta y_2 \\ \theta z_2 \\ dx_2 \\ dy_2 \\ dz_2 \end{vmatrix}
$$

NOTATION : E and G are the usual elastic moduli, A is the cross sectional area, I_y, I_z, J are the sectional properties and L is the member length.

Fig. 3.2 Member Stiffness Matrix.

of its type. However, it does enable an elastic analysis of large, three-dimensional, multi-storey frames to be obtained fairly quickly using machines without recourse to backing store i.e. discs or magnetic tapes. But a program of this kind cannot be used to analyse frames incorporating other structural elements, other than beams and columns that is, such as floors and walls. For this it is necessary to have a more complex program. However, it may be possible to obtain sufficient information of the required accuracy using the orthogonal program by accounting for the behaviour of the floor or other structural element in the stiffness of the beam or column elements. It is mainly for this purpose that the program has been written.

The program is not thought to be too restrictive, since nowadays a very large number of structures are orthogonal in nature; for example most multi-storey buildings are of this kind. A quick appraisal of such a building's elastic behaviour in the form of nodal displacements and member end forces can therefore be obtained by the use of this program.

3.2 Analysis

In a space frame each node (joint) of the structure will have six degrees of freedom, compared to only three in the plane frame case. These degrees of freedom are illustrated in Fig. 3.1(a) by means of the six displacements. The displacements are three deflections; dx, dy, dz and three rotations; θx, θy, θz. Because of this increase in the number of displacements, the

size of the resulting structure stiffness matrix will be four times larger than that in the plane frame case.

The whole process of constructing the structure stiffness matrix can be summarised as follows:-

(i) Form the member stiffness matrices - k.

(ii) Transform these matrices to structure coordinates - AkA^T.

A is the transformation matrix.

(iii) Add the transformed matrices to form the structure stiffness matrix - K.

Or, $$K = \sum_{i=1}^{n} A_i k_i A_i^T$$ where n is the number of members.

The equation shown in Fig 3.2 for the member force/displacement relationship can be rewritten in partitioned form as

$$\begin{vmatrix} P_1 \\ P_2 \end{vmatrix} = \begin{vmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{vmatrix} \begin{vmatrix} d_1 \\ d_2 \end{vmatrix} \qquad (3.1)$$

The matrix A, necessary to transform from member forces to structure forces, is of the form

$$\begin{vmatrix} M_{x'} \\ M_{y'} \\ M_{z'} \\ F_{x'} \\ F_{y'} \\ F_{z'} \end{vmatrix} = \begin{vmatrix} T_{x'x} & T_{x'y} & T_{x'z} & 0 & 0 & 0 \\ T_{y'x} & T_{y'y} & T_{y'z} & 0 & 0 & 0 \\ T_{z'x} & T_{z'y} & T_{z'z} & 0 & 0 & 0 \\ 0 & 0 & 0 & T_{x'x} & T_{x'y} & T_{x'z} \\ 0 & 0 & 0 & T_{y'x} & T_{y'y} & T_{y'z} \\ 0 & 0 & 0 & T_{z'x} & T_{z'y} & T_{z'z} \end{vmatrix} \begin{vmatrix} M_x \\ M_y \\ M_z \\ F_x \\ F_y \\ F_z \end{vmatrix}$$

Fig. 3.3 Force Transformation.

where $T_{x'x}$ is Cos X'OX etc.

Therefore we have

$$P_s = A P_m \tag{3.2}$$

where P is the force vector and where s relates to the structure and m to the member.

From the member force/displacement relationship we have

$$P_m = k d_m \tag{3.3}$$

where d is the displacement vector.

Using the principal of contragredience the displacement transformation is

$$d_m = A^T d_s \tag{3.4}$$

Substituting equations (3.3) and (3.4) into equation (3.2) we arrive at the result

$$P_s = A k A^T d_s \tag{3.5}$$

Using the partitioning as shown in equation (3.1) we can expand equation (3.5) as follows

$$\begin{vmatrix} P_{1s} \\ P_{2s} \end{vmatrix} = \begin{vmatrix} T & 0 \\ 0 & T \end{vmatrix} \begin{vmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{vmatrix} \begin{vmatrix} T^T & 0 \\ 0 & T^T \end{vmatrix} \begin{vmatrix} d_{1s} \\ d_{2s} \end{vmatrix} \tag{3.6}$$

which can be simplified to

$$\begin{vmatrix} P_{1s} \\ P_{2s} \end{vmatrix} = \begin{vmatrix} Tk_{11}T^T & Tk_{12}T^T \\ Tk_{21}T^T & Tk_{22}T^T \end{vmatrix} \begin{vmatrix} d_{1s} \\ d_{2s} \end{vmatrix} \tag{3.7}$$

The matrix T is of the form $\begin{vmatrix} X & 0 \\ 0 & X \end{vmatrix}$ as shown by Fig. 3.3.

Taking the term $Tk_{11}T^T$ as typical of the four submatrices of the stiffness matrix in equation(3.7), we can reduce the formation of $Tk_{11}T^T$ to four 3x3 submatrices of the form

$Xk_{11a}X^T$ giving $Tk_{11}T^T$ as

$$\begin{vmatrix} Xk_{11a}X^T & Xk_{11b}X^T \\ Xk_{11c}X^T & Xk_{11d}X^T \end{vmatrix} \qquad \text{where } k_{11} = \begin{vmatrix} k_{11a} & k_{11b} \\ k_{11c} & k_{11d} \end{vmatrix}$$

Similarly the other terms in equation (3.7) can be expanded so that the whole transformation process can be performed on sixteen 3x3 submatrices using just a 3x3 transformation matrix plus its transpose. This is, in fact, the method used in the program presented here. A simplified flowchart for the program is shown in Fig. 3.6.

3.3 Program.

In the program a procedure called TRANSFORM performs the necessary transformation from member to structure coordinates for each member matrix. Since the member axes are orthogonal to the structure axes the 3x3 transformation matrix will consist of either 0 or ± 1. This matrix is assembled depending on which axis the member's longitudinal axis lies. The member stiffness matrix is previously obtained from the MEMBER procedure.

Having obtained the transformed member stiffness matrix, it is now required to assemble it into the structure stiffness matrix. This is done via the usual node number method. The numbering of the node ends of the member indicates the positions of the member submatrices in the structure stiffness matrix. Fig. 3.4 shows these positions for a member with node numbers i, j. However, it must be remembered that there are six nodal variables. Therefore the actual position of the first element

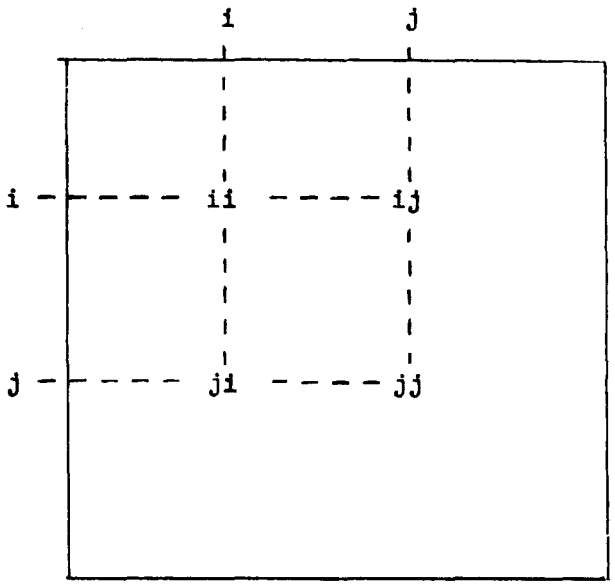
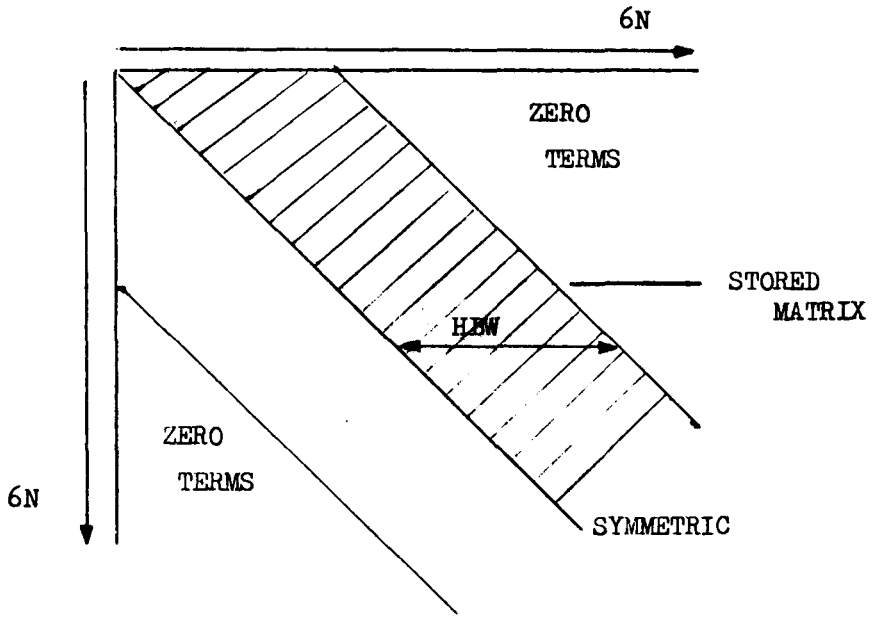


Fig. 3.4 Positions of member submatrices in structure stiffness matrix.



For N nodes the matrix size is 6N x 6N.

Fig. 3.5 Banded Symmetric Structure Stiffness Matrix.

of ii (6x6) is 6x(i-1)+1 and so on for the other elements.

Since the structure stiffness matrix is symmetric and banded the actual stored matrix is far smaller than the whole stiffness matrix. The stored matrix is illustrated in Fig. 3.5, where the important variable becomes the half bandwidth which is dependent on the nodal connectivity of the structure. In fact it depends on the greatest difference between the numbering of connected nodes. If the two nodes concerned are i and j then the half bandwidth is as follows:-

$$HBW = 6x(\ |MAX(i-j)|\ +1) \qquad (3.8)$$

the six occurring because of the six nodal variables. It is well to note that careful numbering of a structure can produce an optimisation in size of the half bandwidth which controls the size of the stored matrix.

In the program the assembling of transformed member stiffness matrices is performed by a procedure called STIFMAT which produces the required structure stiffness matrix in the 'STORED MATRIX' form. The resulting matrix is then triangularised using the usual Gaussian elimination technique.

Restrained variables, i.e. for fixed supports etc., are included via adjustment to the structure stiffness matrix in STIFMAT. The appropriate diagonal element is multiplied by a large number so that its related variable becomes the dominating one.

e.g. to restrain movement of x_1 (i.e. fix $x_1 = 0$).
If the first equation of the force/displacement relationship is as follows:-

$$a_1x_1 + a_2x_2 + a_3x_3 = 0, \qquad (3.9)$$

then

putting a_1x_1 multiplied by 10^{20} gives:-

$$10^{20}a_1x_1 + a_2x_2 + a_3x_3 = 0 \qquad (3.10)$$

where $10^{20}a_1x_1 \gg a_2x_2,\ a_3x_3$. Therefore equation (3.10) approximates to

$$10^{20}a_1x_1 = 0 \qquad \text{i.e.} \quad x_1 = 0.$$

A similar procedure is employed for other variables which require restraining.

For each load case the deflections are obtained via the usual backsubstitution operation on the triangularised matrix. The member forces are output with respect to the member axes. In order to do this a procedure FORCE is brought into action. This procedure will produce the end forces and moments of a member. Therefore, having obtained the global displacements, we require the member forces as follows:-

$$\begin{vmatrix} P_{1m} \\ P_{2m} \end{vmatrix} = \begin{vmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{vmatrix} \begin{vmatrix} T^T & 0 \\ 0 & T^T \end{vmatrix} \begin{vmatrix} d_{1s} \\ d_{2s} \end{vmatrix} \qquad (3.11)$$

or more compactly,

$$P_m = k\, A^T d_s \qquad (3.12)$$

The input and output of the program is very basic and has not been developed for multi-user operation; member properties - area, inertia etc.- are input with each member. Loading is only applied at the nodes; distributed loading is not handled directly but of course this type of loading can be reduced to nodal loading only. The program has no specific units built into it so that in can be versatile; however, it must be remembered that all input quantities must be consistent.

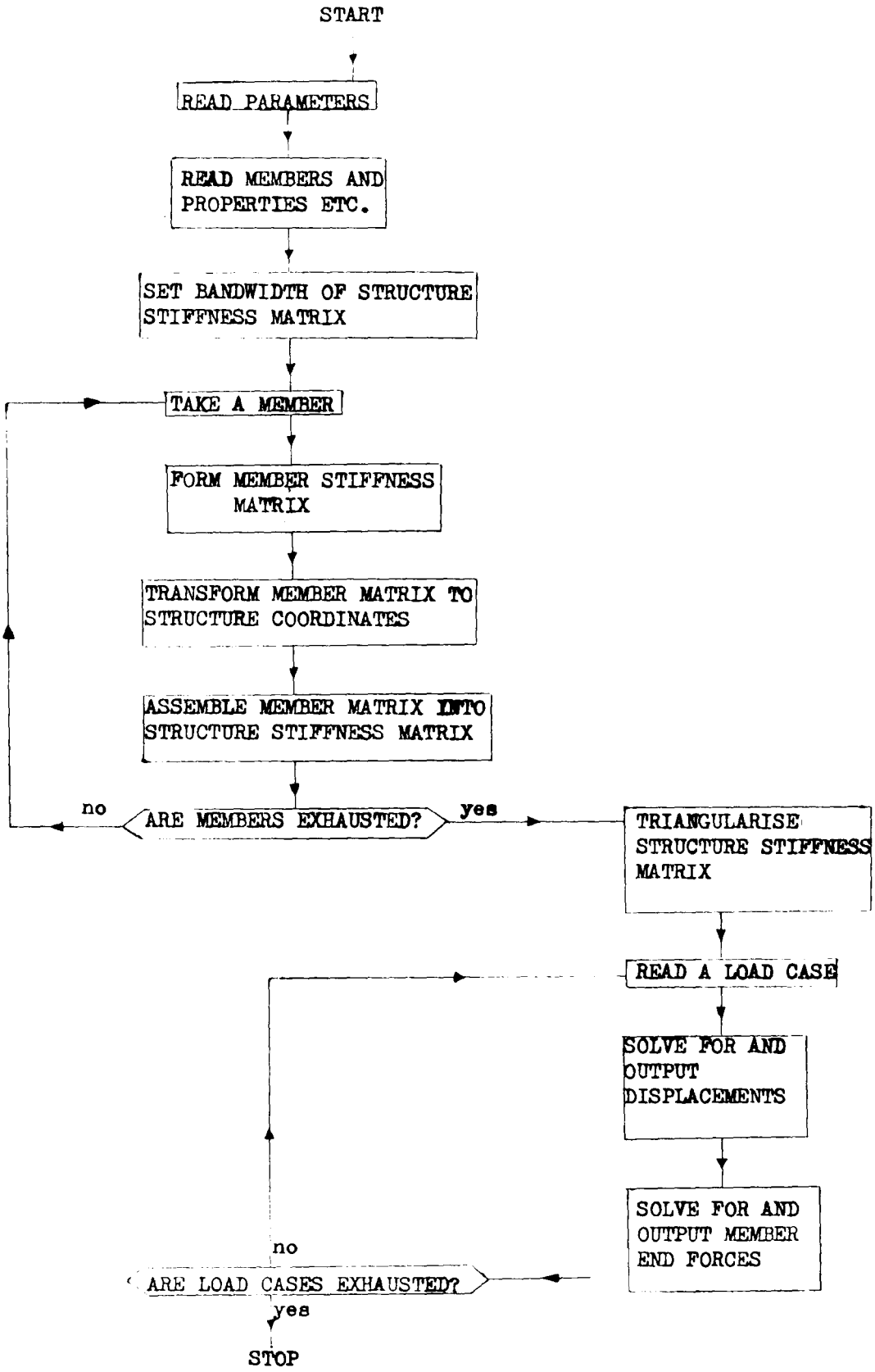
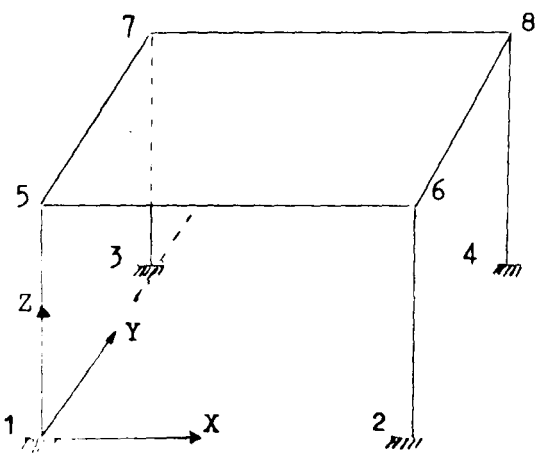


Fig. 3.6 Flowchart for Rigid Orthogonal Skeletal Space Frame Analysis Program.

3.4 Example Illustrating the Use of Rigid Orthogonal Space Frame Program.

A simple three-dimensional frame, consisting of column and beam elements, has been analysed using the program described in this section. No discussion on the results obtained from this example will be undertaken. Both input and output data are given.

Example: Three-dimensional analysis of a table-like frame under various loadings.



Sketch of Frame Showing Joint Numbering.

UNITS USED:- KN, mms All input/output in these units.

1. INPUT DATA.

(a) Material Constants - Young's Modulus (E) Shear Modulus (G)

13.78 5.512

(b) Parameters - Restraints Joints Members Load Cases

24 8 8 4

(c) Restraints - Joint Type of Restraint Joint Type of Restraint

1	1	2	1
1	2	2	2
1	3	2	3
1	4	2	4
1	5	2	5
1	6	2	6
3	1	4	1
3	2	4	2
3	3	4	3
3	4	4	4
3	5	4	5
3	6	4	6

(d) Member Properties -

Member	Area	IY	IZ	J
1 - 5	40000	$1.33_{10}8$	$1.33_{10}8$	$2.66_{10}8$
2 - 6	40000	$1.33_{10}8$	$1.33_{10}8$	$2.66_{10}8$
3 - 7	40000	$1.33_{10}8$	$1.33_{10}8$	$2.66_{10}8$
4 - 8	40000	$1.33_{10}8$	$1.33_{10}8$	$2.66_{10}8$
5 - 6	64000	$5.46_{10}8$	$5.46_{10}8$	$1.092_{10}9$
5 - 7	64000	$5.46_{10}8$	$5.46_{10}8$	$1.092_{10}9$
6 - 8	64000	$5.46_{10}8$	$5.46_{10}8$	$1.092_{10}9$
7 - 8	64000	$5.46_{10}8$	$5.46_{10}8$	$1.092_{10}9$

(e) Structure Coordinates -

XC	YC	ZC
0	0	0
6000	0	0
0	6000	0
6000	6000	0
0	0	6000
6000	0	6000
0	6000	6000
6000	6000	6000

(f) Load Case 1 * - Loads 13

Joint	Value	Direction
8	-2	5
5	-9000	1
5	9000	2
5	-9	6
6	-9000	1
6	-9000	2
6	-9	6
7	9000	1
7	9000	2
7	-9	6
8	9000	1
8	-9000	2
8	-9	6

(g) Load Case 2 * - Loads 14

Joint	Value	Direction
5	-15000	1
5	15000	2
5	-15	6
6	-15000	1
6	-15000	2
6	-15	6
7	15000	1
7	15000	2
7	-15	6
8	15000	1
8	-15000	2
8	-15	6
5	5	4
7	5	4

END OF INPUT DATA

* A load vector applied to joints 5, 6, 7, 8 is derived from the formulae given by Zienkiewicz[1] for a uniformly distributed load between the stated joints.

2. OUTPUT.

(a) Results of Load Case 1.

(i) Structure joint displacement. - All nodal displacements are given in TABLE 3.1 where the deflections are in mms.

(ii) Member-end forces. - Member forces are presented in TABLE 3.2 where the forces are in KNs and the moments in KNmms.

TABLE 3.1.

Joint	ROTX	ROTY	ROTZ	XDIR	YDIR	ZDIR
1	0.0000	-0.0000	-0.0000	-0.0000	-0.0000	-0.0000
2	0.0000	0.0000	-0.0000	0.0000	-0.0000	-0.0000
3	-0.0000	-0.0000	-0.0000	-0.0000	0.0000	-0.0000
4	0.0000	0.0000	-0.0000	0.0000	-0.0000	-0.0000
5	-0.0023	0.0024	-0.0008	-2.0758	-2.8714	-0.1028
6	-0.0021	-0.0025	-0.0008	-2.0808	-8.1011	-0.1036
7	0.0025	0.0025	-0.0008	2.0808	-2.8765	-0.0932
8	0.0027	-0.0024	-0.0008	2.0758	-8.1129	-0.0923

TABLE 3.2.

Member	MX1	MY1	MZ1	AX1	SY1	SZ1
1 - 5	188.05	2072.32	2278.80	9.44	0.99	-0.93
2 - 6	188.05	-874.67	3785.40	9.52	1.48	0.54
3 - 7	188.25	874.67	-668.26	8.56	-0.48	-0.54
4 - 8	188.25	-2072.32	840.20	8.48	0.01	0.93
5 - 6	-149.15	5607.69	767.83	0.74	0.26	0.15
5 - 7	-118.27	5170.35	-579.78	0.74	-0.19	0.29
6 - 8	-118.27	4052.89	-579.78	1.74	-0.19	0.67
7 - 8	-149.51	6496.78	770.11	0.74	0.26	-0.15

Member	MX2	MY2	MZ2	AX2	SY2	SZ2
1 - 5	-188.05	3510.57	3680.49	-9.44	-0.99	0.93
2 - 6	-188.05	-2384.95	5096.26	-9.52	-1.48	-0.54
3 - 7	-188.25	2384.95	-2215.15	-8.56	0.48	0.54
4 - 8	-188.25	-3510.57	-797.74	-8.48	-0.01	-0.93
5 - 6	149.15	-6496.78	767.83	-0.74	-0.26	-0.15
5 - 7	118.27	-6934.37	-581.86	-0.74	0.19	-0.29
6 - 8	118.27	-8052.74	-581.86	-1.74	0.19	-0.67
7 - 8	149.51	-5607.69	770.11	-0.74	-0.26	0.15

(b) Results of Load Case 2.

(i) Structure joint displacement. - All nodal displacements are given in TABLE 3.3 where the deflections are in mms.

(ii) Member-end forces. - Member forces are presented in TABLE 3.4 where the forces are in KNs and the moments in KNmms.

TABLE 3.3.

Joint	ROTX	ROTY	ROTZ	XDIR	YDIR	ZDIR
1	0.0000	0.0000	-0.0000	0.0000	-0.0000	-0.0000
2	0.0000	0.0000	-0.0000	0.0000	-0.0000	-0.0000
3	-0.0000	0.0000	-0.0000	0.0000	0.0000	-0.0000
4	-0.0000	0.0000	-0.0000	0.0000	0.0000	-0.0000
5	-0.0040	0.0050	-0.0000	27.4650	0.0042	-0.1371
6	-0.0040	-0.0031	-0.0000	27.4397	0.0042	-0.1894
7	0.0040	0.0050	-0.0000	27.4650	-0.0042	-0.1371
8	0.0040	-0.0031	-0.0000	27.4397	-0.0042	-0.1894

TABLE 3.4.

Member	MX1	MY1	MZ1	AX1	SY1	SZ1
1 - 5	0.00	-5341.51	2455.83	12.60	1.23	1.27
2 - 6	0.00	-10248.82	2455.83	17.40	1.23	3.73
3 - 7	0.00	-5341.51	-2455.83	12.60	-1.23	1.27
4 - 8	0.00	-10248.82	-2455.83	17.40	-1.23	3.73
5 - 6	-0.00	17293.65	-0.00	3.73	-0.00	-2.40
5 - 7	-0.00	10087.06	0.00	1.23	0.00	-0.00
6 - 8	-0.00	10087.06	0.00	1.23	0.00	0.00
7 - 8	-0.00	17293.65	-0.00	3.73	-0.00	-2.40

Member	MX2	MY2	MZ2	AX2	SY2	SZ2
1 - 5	-0.00	-2293.65	4912.94	-12.60	-1.23	-1.27
2 - 6	-0.00	-12116.02	4912.94	-17.40	-1.23	-3.73
3 - 7	-0.00	-2293.65	-4912.94	-12.60	1.23	-1.27
4 - 8	-0.00	-12116.02	-4912.94	-17.40	1.23	-3.73
5 - 6	0.00	-2883.98	-0.00	-3.73	0.00	2.40
5 - 7	0.00	-10087.06	0.00	-1.23	-0.00	0.00
6 - 8	0.00	-10087.06	0.00	-1.23	-0.00	-0.00
7 - 8	0.00	-2883.98	-0.00	-3.73	0.00	2.40

END OF OUTPUT

The total time taken for both load cases was 30 seconds and a total of 13k storage locations were required, the program being run on an Elliot 4130 machine.

The validity of results obtained from this program was examined by checking plane frame solutions using the 'Tee-beam' program and by comparison with known results including those from a test structure given by Majid & Williamson[2].

3.5 References.

1. O. C. Zienkiewicz - 'The Finite Element Method in Engineering Science'. p 184, Table 10.3, McGraw Hill London 1971.
2. K. I. Majid & M. Williamson - 'Linear Analysis of Complete Structures by Computers'. Proceedings of I.C.E. Vol. 38, Oct. 1967, 247-266.

4. USE OF FINITE ELEMENTS FOR SLAB STIFFNESS REPRESENTATION.

4.1 Introduction.

For completion of the multi-storey frame program, some method of floor analysis was required. The problem of introducing the structural effect of floor slabs on the rest of the frame was to some extent quite easy, since various methods of slab solution were available. Thus the problem was essentially one of choosing the most suitable of these methods.

In order to make a decision it was well to be aware of the requirements of the frame analysis and also to bear in mind which slab solution would be the most advantageous in this respect. The frame analysis is based on an elastic stiffness matrix force/displacement method.

The more well known methods of slab or thin plate analysis are extensively documented. To summarise these a list of the most prominent ones is now presented together with comments on their suitability.

1) British Standard empirical method for design purposes[1].

Does not yield a force/displacement relationship as required.

2) Yield Line Analysis[2,3].

It is not an elastic analysis and therefore not compatible with the frame analysis.

3) Force/displacement relationships - elastic.

Several techniques are available to provide

these relationships but the mathematical solution is an approximation of the theoretical equations. These equations in turn are not truly representative of material behaviour except under certain circumstances.

One of the simplest methods of analysis for plate flexure is by means of a finite difference solution of the biharmonic equation. More sophisticated and extensive procedures have been developed, for example the finite element and localised Ritz techniques[7,8]. Other methods, including model analysis and beam analogy, are further possibilities.

It is quite evident that one of this last group would be most consistent with the stiffness equations of the frame and it seems logical to choose the most accurate of the methods and the most attractive to the main program. This is obviously the finite element technique which would provide an elastic solution in the form of force/displacement relationships resulting in the use of the usual stiffness matrix type of equations. Not only would the method provide the flexural mode but also that for shear i.e. the plane stress case.

4.2 Plate Flexure.

A rectangular finite element, illustrated in Fig. 4.1, is used to establish the stiffness matrix for plate bending. This element shape is chosen since it is the easiest to handle and also because it fits well with the orthogonal nature of the space frame.

The full mathematical derivation of the finite element matrix will not be presented here since an extensive coverage of

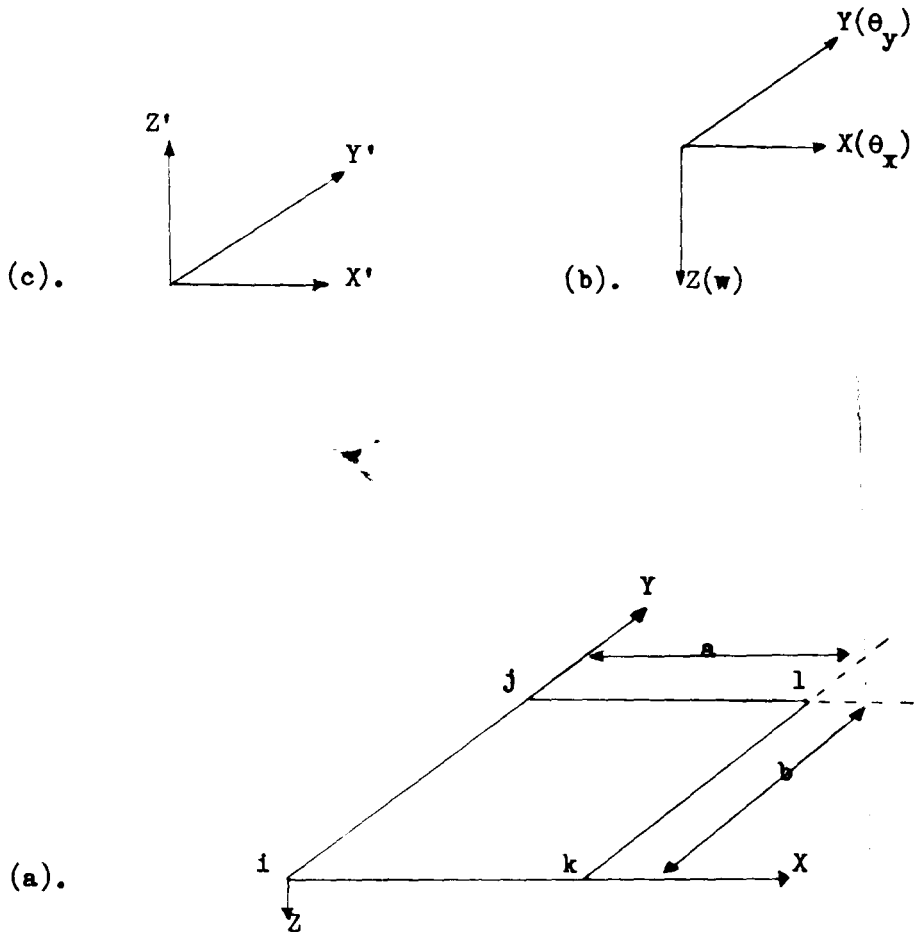


Fig. 4.1. (a). Rectangular finite element.

Element axes (b) and structure axes (c).

this is given by Zienkiewicz[4]. However, the basic concepts and relationships will be stated so as to give the reader the necessary information to understand how the element is used in the frame program.

The element derivation is based on classical plate theory and therefore is subject to the same limitations i.e. thin plates and small deflections to ensure linear variation of stresses and strains on lines normal to the plane of the plate.

It is known that the state of strain of the plate can be completely described by one variable, this being w, the lateral displacement of the 'middle plane' of the plate. A shape function - a polynomial defining w - is chosen so that continuity of w along boundaries is ensured. However, the slope of w across boundaries is not continuous, the function being a 'non-conforming' one

$$w = a_1 + a_2x + a_3y + a_4x^2 + a_5xy + a_6y^2 + a_7x^3 + a_8x^2y + a_9xy^2 + a_{10}y^3 + a_{11}x^3y + a_{12}xy^3. \qquad (4.1)$$

For a full discussion on conforming and non-conforming shape functions the reader is again referred to Zienkiewicz.

The polynomial is chosen with twelve unknown parameters since the element possesses twelve degrees of freedom. This is illustrated by the three displacements per node as shown in Fig. 4.1.

The element displacement vector is of the form

$$d = \left| d_i \; d_j \; d_k \; d_l \right|^T \qquad (4.2)$$

where $d_i = \begin{vmatrix} w_i \\ \theta_{xi} \\ \theta_{yi} \end{vmatrix} = \begin{vmatrix} w \\ -dw/dy \\ dw/dx \end{vmatrix}_i$ (4.3)

The unknown parameters are evaluated using the nodal displacements and substituting in the nodal coordinates for the x and y where appropriate. For example, for node i in Fig. 4.1,

$$w_i = a_1$$
$$-dw/dy_i = \theta_{xi} = -a_3$$
$$\theta_{yi} = a_2 \quad \text{etc.}$$

For the whole element, 12 equations are obtained, which can be expressed as:-

$$d = C\,a \tag{4.4}$$

where $a = \begin{vmatrix} a_1 & a_2 & a_3 \cdots\cdots & a_{12} \end{vmatrix}^T$

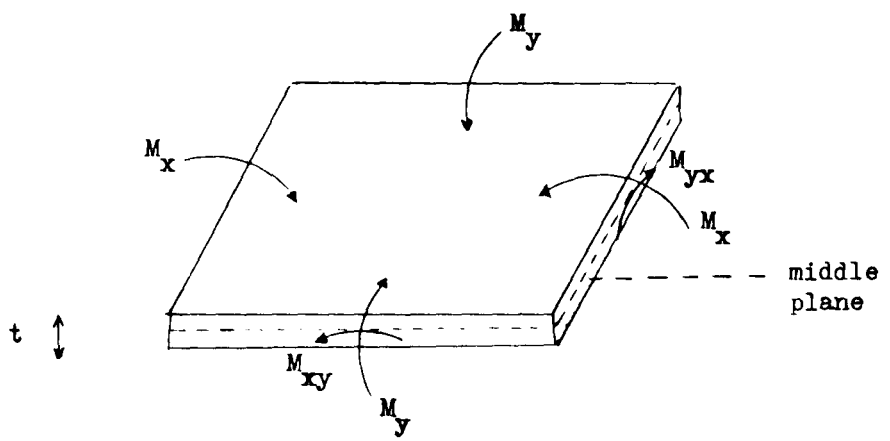


Fig. 4.2 Bending moments on finite element.

t - plate thickness.

M_x - moment per unit length in x direction.

M_y - moment per unit length in y direction.

M_{xy} - twisting moment per unit length.

The element moment vector is as follows:-

$$M = \left| M_i \quad M_j \quad M_k \quad M_l \right|^T \tag{4.5}$$

where $M_i = \left| \begin{matrix} M_x \\ M_y \\ M_{xy} \end{matrix} \right|_i$ for node i.

These moments are related to the element curvatures or strains in the usual way,

$$\text{i.e.} \quad M_x = -D_x \left(\frac{d^2w}{dx^2} + \nu \frac{d^2w}{dy^2} \right) \text{ etc.} \tag{4.6}$$

D_x is the flexural rigidity.

Thus arranging in the more compact matrix form, we have

$$M = D.\underline{E}$$

$$\text{where} \quad \underline{E} = \left| \begin{matrix} -d^2w/dx^2 \\ -d^2w/dy^2 \\ 2d^2w/dxdy \end{matrix} \right| \quad \text{the strain vector} \tag{4.7}$$

D is the elasticity matrix.

The program was restricted to isotropic slabs and therefore in this case the elasticity matrix will be

$$D = Et^3/12(1-\nu^2). \left| \begin{matrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & (1-\nu)/2 \end{matrix} \right| \tag{4.8}$$

E Young's Modulus, ν Poisson's Ratio.

The strain matrix can also be expressed in terms of the nodal displacements in matrix form as follows:-

$$\underline{E} = Q.a = Q.C^{-1}d \tag{4.9}$$

Therefore deriving the finite element matrix in the usual way by equating internal and external work, the expression for the stiffness matrix is obtained

$$k = \left| C^{-1} \right|^T \left[\iint Q^T.D.Q \; dx \; dy \right] \left| C^{-1} \right| \tag{4.10}$$

where the force/displacement relationship is the usual form $F = k.d$ and the internal moments are given by

$$M = k' d \tag{4.11}$$

where $$k' = D.Q.C^{-1} \tag{4.12}$$

The explicit form of these matrices, k and k', are given by Zienkiewicz[4].

4.3 Plate shear - Plane stress.

In order to be consistent with the flexural element, the plane stress element was also made rectangular. The usual shape for this element is triangular since it provides a more flexible element to use for irregular shaped slabs. However, the rectangular element does have the advantage that because of the increase in degrees of freedom compared to the triangular one, a more accurate representation of the varying stress field is possible.

The state of displacement of a plate can be specified by two quantities, namely, u and v, the deflection in the x and y direction respectively. These deflections are illustrated in Fig.4.3. Thus the shape functions were chosen as follows:-

$$u = a_1 + a_2x + a_3y + a_4xy.$$
$$v = a_5 + a_6x + a_7y + a_8xy. \tag{4.13}$$

so that the strains in the plate would be linear functions of x and y. The procedure for determining the stiffness matrix is similar to that used for the previous element.

The evaluation of the unknown parameters is understood as

$$a = C^{-1}d \tag{4.14}$$

where $a = \left| a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7\ a_8 \right|^T$

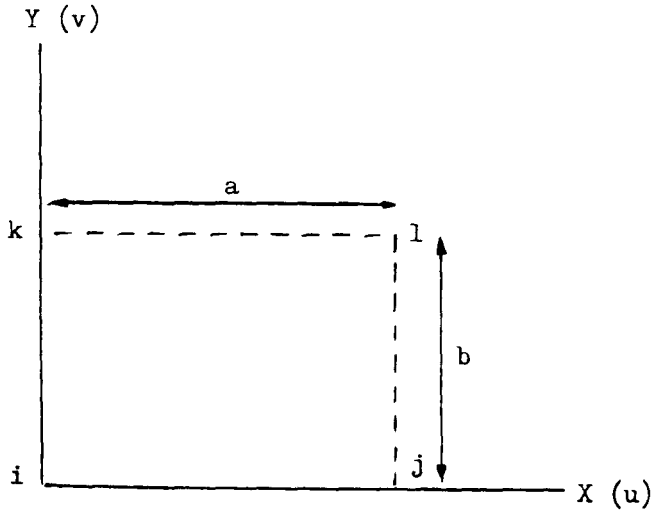


Fig. 4.3 Plane stress finite element with axes and displacements.

and $d = \left| d_i \; d_j \; d_k \; d_l \right|^T$ where $d_i = \begin{vmatrix} u_i \\ v_i \end{vmatrix}$

The strains in the element are specified as follows:-

$$\underline{E} = \left| E_x \quad E_y \quad E_{xy} \right|^T$$ where $\underline{E}$ is the strain vector and

$$\begin{vmatrix} E_x \\ E_y \\ E_{xy} \end{vmatrix} = \begin{vmatrix} du/dx \\ dv/dy \\ (du/dy + dv/dx) \end{vmatrix}$$

This strain vector can be rewritten in terms of the unknown parameters a_i where

$$\underline{E} = Q.a \qquad (4.15)$$

and using equation (4.14) the previous equation can be expressed in terms of the nodal deflections d.

$$\underline{E} = Q.C^{-1}d \qquad (4.16)$$

As with the flexural element it is possible to express the internal stresses of the element in terms of its state of strain.

i.e.

$$\begin{vmatrix} S_x \\ S_y \\ S_{xy} \end{vmatrix} = \frac{E}{1-\nu^2} \begin{vmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & (1-\nu)/2 \end{vmatrix} \begin{vmatrix} E_x \\ E_y \\ E_{xy} \end{vmatrix} \qquad (4.17)$$

S_x, S_y are the direct stresses.

S_{xy} is the shear stress.

ν is Poisson's ratio and E is Young's modulus.

or in more compact matrix form

$$S = D.\underline{E} \qquad (4.18)$$

where D is the elasticity matrix.

Now by equating internal and external work the required stiffness matrix is obtained as

$$K = \left|C^{-1}\right|^T \left[\iiint_{VOLUME} Q^T.D.Q \; dvol. \right] \left|C^{-1}\right| \qquad (4.19)$$

The explicit form of this matrix is reasonably easy to obtain and it is given by Jenkins[9].

4.4 Examples and Results Illustrating the Use of Finite Elements.

This brief description of the type of information available by utilisation of the finite elements previously described is not intended to be a thorough exposé on the merits of the finite element technique for thin plate analysis. Its purpose is to illustrate the properties of the two elements discussed, to give an example of the kind of information which is available and also to give some insight into the accuracy and convergence of results with respect to the mesh size of the elements. For a detailed description of finite element applications the reader is referred to references 4 and 6 where the subject is covered extensively. The examples given here are for the analysis of a square plate for both flexure and plane stress. Additional results are given to demonstrate accuracy and convergence.

Plate Flexure.

Fig. 4.4 shows the displacements of a square plate loaded for bending. The results were obtained via a 6x6 finite element analysis using the element described in 4.2. The numerical output form of the results is illustrated together with a contour plot derived from just one set of nodal displacements, namely the vertical deflection w.

The internal bending moments for the same plate as described above are displayed in Fig. 4.5. and are derived from the same analysis. The contour plot shown is for the bending moment per unit length along the X-axis, M_x. The stress is highly concentrated around the corners at which the plate is fixed.

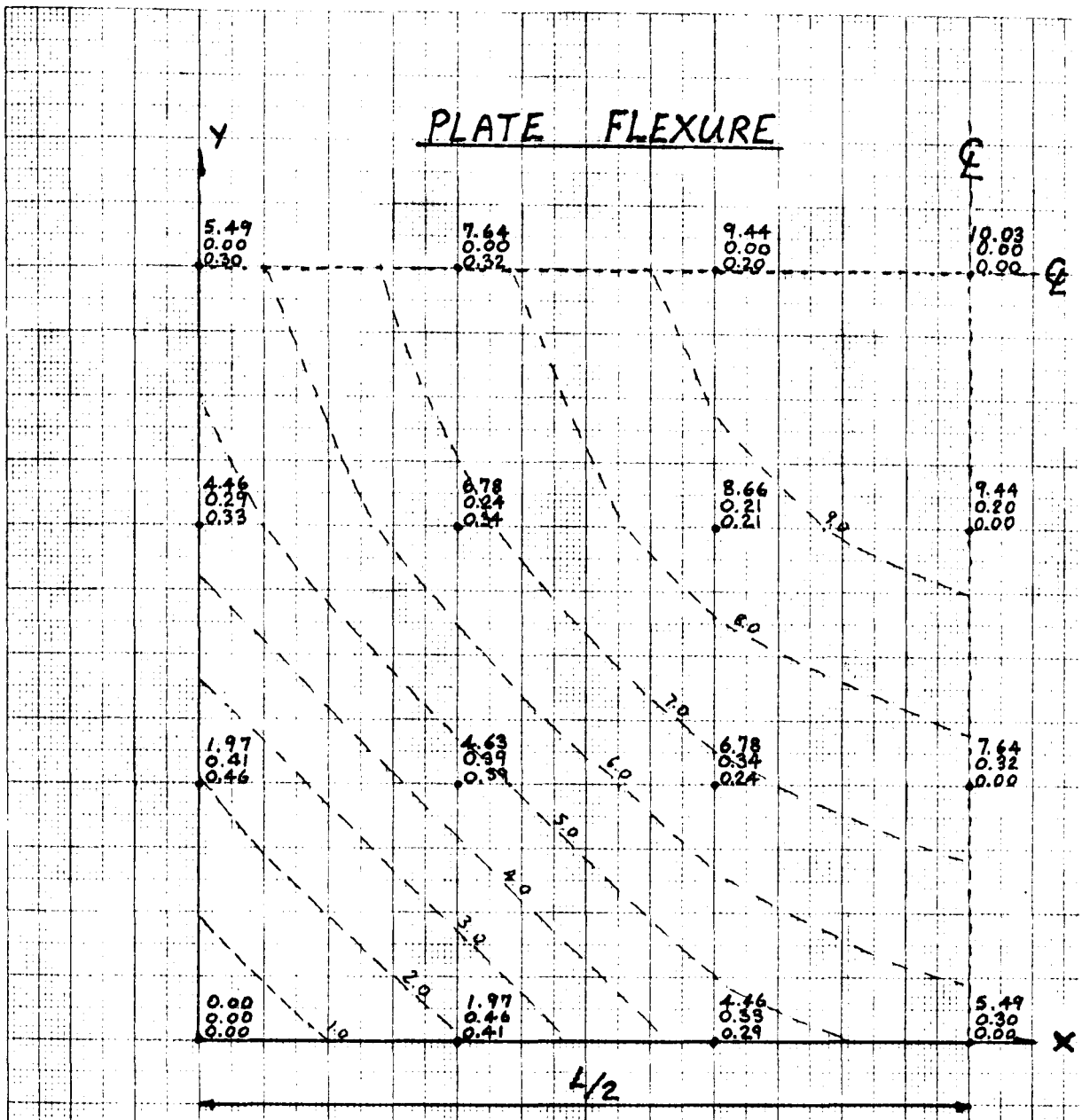


Fig. 4.4 Displacement of a uniformly loaded square plate fixed rigidly at each corner.

Displacements at mesh nodes a). ω. b) θ_x c) θ_y.

displacements : vertical deflection = $\omega\, q\, L^4 / D \times 10^{-3}$

rotation = $\theta\, q\, L^4 / D \times 10^{-3}$

D Flexural rigidity L length of side of plate q load per unit area

Contours shown are for vertical displacement only.

A 6.6 finite element was used.

PLATE FLEXURE

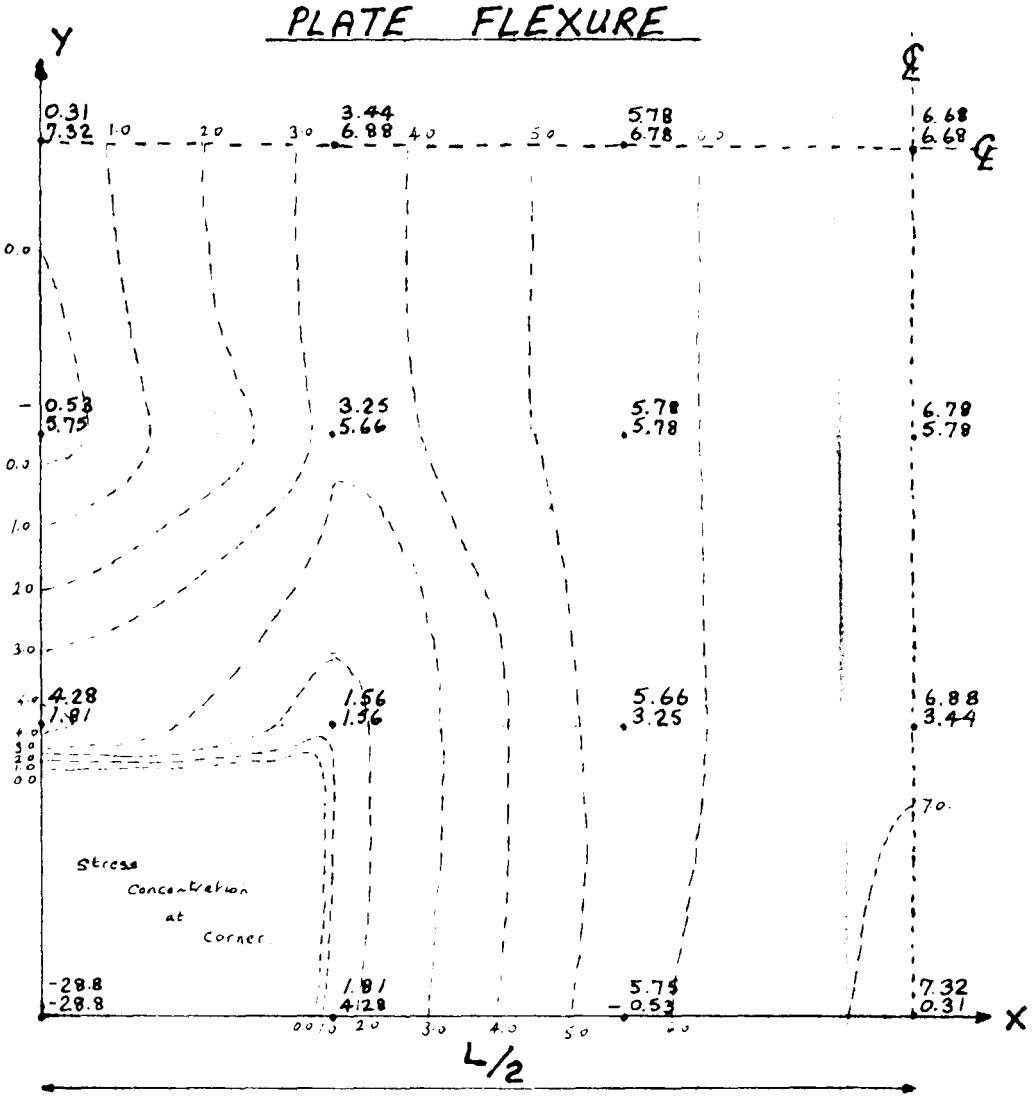


Fig. 4.5 Slab Internal Moments for a uniformly loaded square slab fixed rigidly at each corner.

Moments at mesh node a) M_x b) M_y.

moments per unit length : moment = $M_x \cdot (10^2/qL^2)^{-1}$

q = load L size of plate D flexural rigidity.

Contours shown are for M_x only.

A 6 x 6 finite element was used.

PLANE STRESS

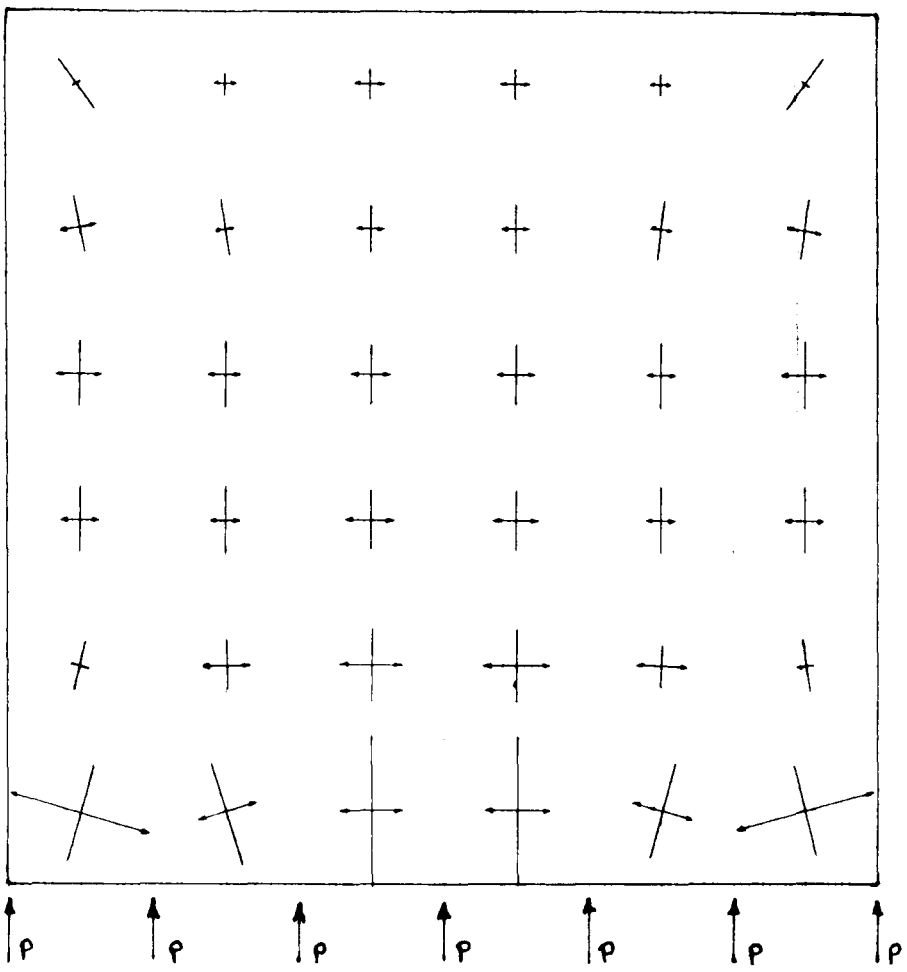


Fig 4.6 Plot of principal stresses of a square slab fully restrained at corners and loaded in shear.

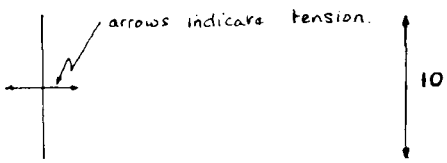
Length of side of slab · 18 metres.

Slab thickness 150 mms.

Elastic Modulus 13·78 KN/mm.²

Poisson's Ratio 0·15

Element Mesh 6 × 6



stresses in KN/mm² × 10^{-6}.

Insufficient information is available from the analysis to plot this area. For more detailed information a finer mesh pattern would be necessary.

Plane Stress.

The example given in Fig. 4.6. shows an alternative representation of the results obtained for element forces. For this plane stress case, the principal stresses and their lines of action have been ploted instead of the directly calculated stresses orientated with the element axis. Note how the stress distribution tends to uniformity towards the centre of the plate.

Accuracy and Convergence.

Fig. 4.7. compares the results for the internal bending moments in a plate obtained from an analysis using the finite element as described in 4.2. with those obtained via a finite difference formulation similar to that of Salonen[5]. It should be noted that close agreement in results is in evidence.

Fig. 4.8. demonstrates the effects of mesh density on the results from a finite element analysis for the vertical displacement across a diagonal of a square plate which is loaded uniformly and fixed rigidly at each corner. It can be seen that even for a relatively small number of elements reasonable agreement with more detailed results will be forthcoming. Also it is useful to notice the convergence of the central deflection with respect to the mesh size.

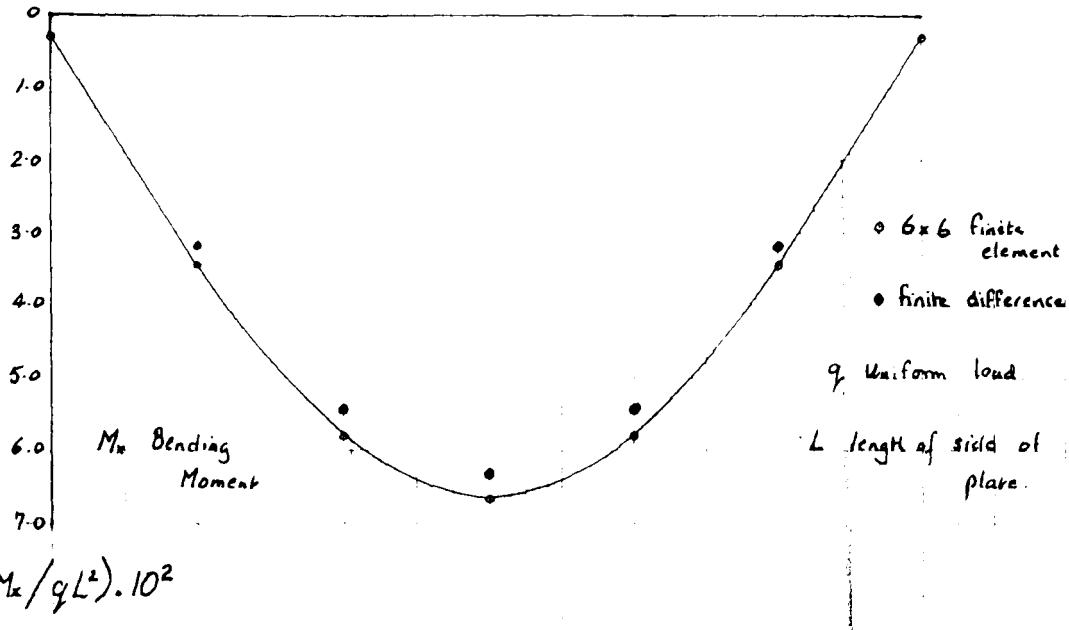


Fig. 4.7 Comparison of internal bending moments from (a) Finite element and (b) finite difference analysis.

Plot of Mx across centre line on x-axis of square plate clamped at each corner.

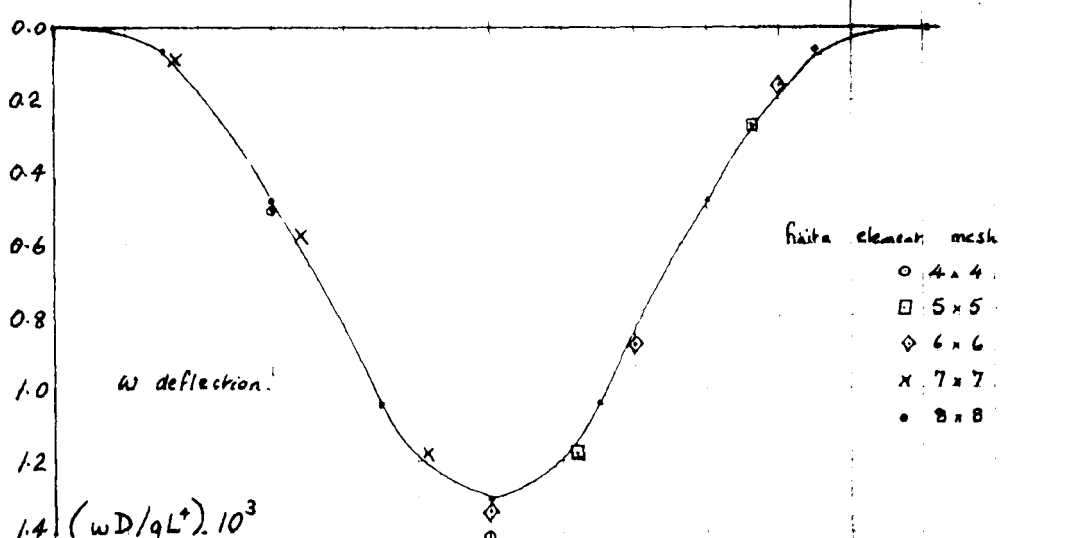


Fig. 4.8 Deflection across diagonal of a square plate clamped on all sides, loaded uniformly.

q uniform load. L length of side D flexural rigidity.

TABLE 4.1 Central deflection of a square plate clamped on all sides and loaded uniformly.

Element Mesh	Central Deflection γ.
2 x 2	1.48
4 x 4	1.41
6 x 6	1.34
8 x 8	1.30
16 x 16	1.28
Exact (Timoshenko)	1.26

A rectangular finite element was used throughout.

Actual deflection = $(\gamma qL^4/D) \times 10^{-3}$.

where γ is from TABLE 4.1 above.

q is uniformly distributed load.

L is length of side of plate.

D is flexural rigidity.

The results given in TABLE 4.1 illustrate the convergence of the finite element results and give an illustration of their accuracy by comparison with the known result for this case, derived by Timoshenko[10].

The examples on convergence and accuracy are by no means proof of the technique but merely demonstrations of its properties. For further information on convergence and accuracy the reader is refered again to more detailed accounts as given in the references.

4.5 References.

1. British Standards Institution (London) CP114:1957 -'Structural Use of Reinforced Concrete in Buildings'. (Reset and reprinted in 1965).

   Note: Superceded lately by CP110 New Unified Code. 'The Code of Practice for Structural Use of Concrete'. B.S.I. (London) 1972.

2. L. L. Jones & R. M. Wood - 'Yeild-line Analysis of Slabs'. Thames & Hudson, Chatto & Windus 1967.

3. R. H. Wood - 'Plastic and Elastic Design of Slabs and Plates (with particular reference to R.C. floor slabs)'. Thames & Hudson London 1961.

4. O. C. Zienkiewicz - 'The Finite Element Method in Structural and Continuum Mechanics'. McGraw Hill 1971.

5. E. M. Salonen - 'A rectangular plate bending element, the use of which is equivalent to the use of the finite difference method'. International Journal of Numerical Methods in Engineering 261-273 Vol. 1 (1969).

6. O. C. Zienkiewicz - 'The Finite Element Method in Engineering Science'. McGraw Hill 1971.

7. J. G. A. Croll & A. C. Walker - 'The Finite Difference and Localised Ritz Methods'. The International Journal for Numerical Methods in Engineering 155-160 Vol 3 (1971).

8. J. G. A. Croll - 'Hermitian Methods for the Approximate Solution of Partial Differential Equations'. J. Inst. Math. Applics. 6. 365-374 (1970).

9. W. M. Jenkins - 'Matrix and Digital Computer Methods in Structural Analysis' McGraw Hill, 1969 144-148.

10. S. Timoshenko & S. Woinowsky-Krieger - 'Theory of Plates and Shells'. McGraw Hill, 2nd ed., 1959.

5. RIGID ORTHOGONAL SPACE FRAME WITH FLOOR DECKS.

5.1 Introduction.

The aim of this program is to produce an elastic analysis for a multi-storey space frame whose floors are considered as slab decks. It is obviously an extension of the skeletal program of section 3, but now it is required to introduce into the program the slab matrices described in section 4. However, the problem still consists of forming and solving the structure stiffness matrix.

The problem is, in the final outcome, identical to that of the skeletal case i.e. to obtain a structure stiffness matrix, however, the method of reaching that final state is not the same. Because of the increased scope of the program a large increase in the size of this matrix is produced resulting in increased demand for computer storage. Therefore to follow the same approach as in section 3 results in an excessive demand for backing store and a high degree of data transfer to and from the backing store during run time.

Thus it would seem that a reduction in the size of the final structure stiffness matrix would benefit the solution of the problem. This program attempts to do this using a 'variable' elimination technique so that the actual working matrices are kept to a manageable size and that data transfers are kept to a minimum. The following section describes the method of solution in detail.

The first subsection outlines the basic flow of the solution

process and is followed by a discussion of the effects of the computer and systems available upon this solution. Then a full description of the program with illustrative flowcharts is presented and finally some examples are given to demonstrate the use and versatility of the developed program - DECK1

5.2 Method of solution.

A solution for the elastic analysis of an orthogonal frame structure with floor decks is obtained from a stiffness matrix analysis. However, with the complexity of the problem, a direct solution process is impracticable and therefore it is reduced to several smaller stages.

Basically floor stiffness matrices are formed independently for a set mesh pattern and the resulting matrix is reduced to involve only the column/floor interacting nodes. This is done for both the flexural and the plane stress cases independently of each other. These 'condensed' floor matrices and vectors are then stored for future use.

The structure matrix is formed for only two floors at a time, starting with the ground floor, and the appropriate 'condensed' floor matrices are added where required. This 'part-structure' matrix is then also 'condensed' so that it refers only to the higher floor level of the two. Resulting matrices are again stored ready for a back-substitution process.

The floor above is then added to the 'condensed' structure matrix and the process repeated until the final floor level is reached, when a full solution is obtained by a 'cascading'

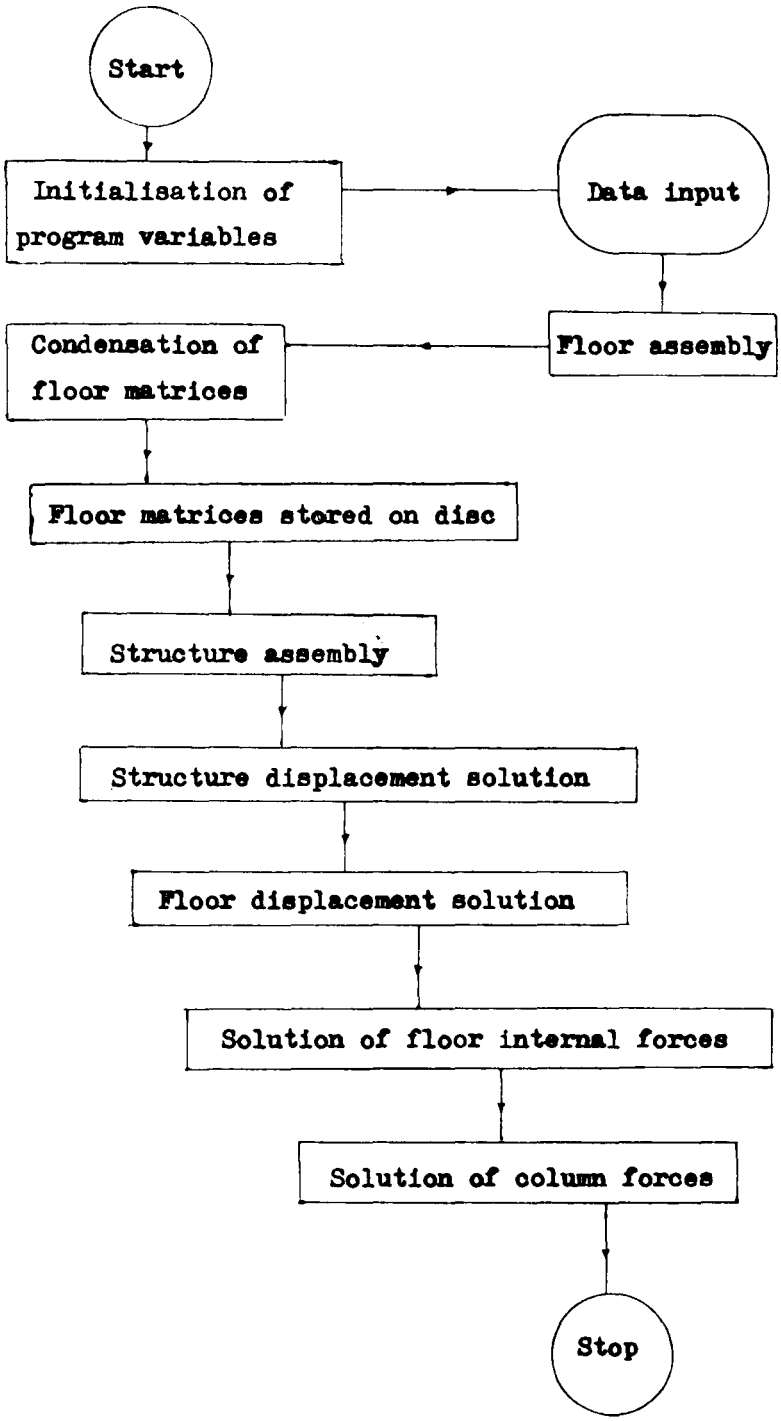


Fig. 5.1 Diagram outlining flow process for the DECK1 program.

back-substitution process.

Slab-supporting beams are accounted for by inclusion in the appropriate floor matrix. The whole process is illustrated in table form in Fig. 5.1.

5.3 Computational problems.

The choice of the method of solution for a framed structure incorporating a slab deck will itself inevitably lead to computational problems. The problems arising in this case reduce to three major ones. These are, not in any order of difficulty, as follows:-

(i) Continuity between slab/beam connections.

(ii) Finite element mesh placing.

(iii) The in-plane rotational stiffness of the slab.

The first and second are directly related and the solution adopted for (ii) removes problem (i). Therefore problem (ii) is dealt with first.

The placing of rectangular finite elements on a floor deck is open to a certain amount of choice and several possibilities were considered. It was thought that it may have been feasible to assume that all columns formed a regular rectangular pattern on a floor deck as in Fig. 5.2. In this case a mesh pattern could be fixed with all elements identical and element size related to dimensions 'a' and 'b'. This would allow the automatic setting up of the finite element pattern but would create two rather serious restrictions. It would restrict the column placing so that only rectangular patterns could be used and it would prevent

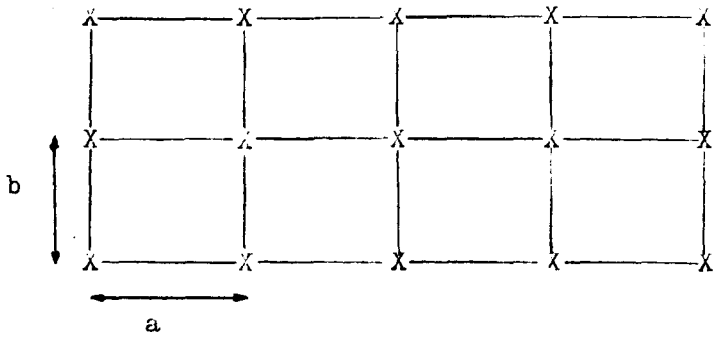


Fig. 5.2 Regular column pattern on a floor.

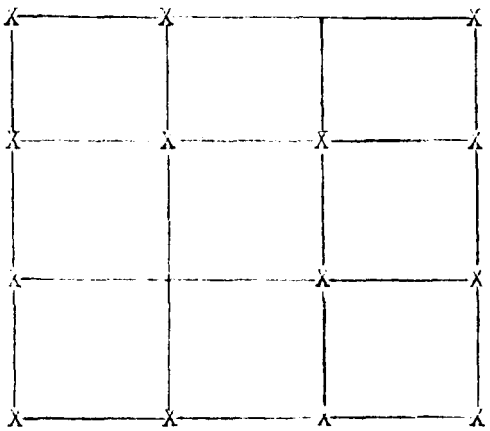


Fig. 5.3 Irregular column pattern.

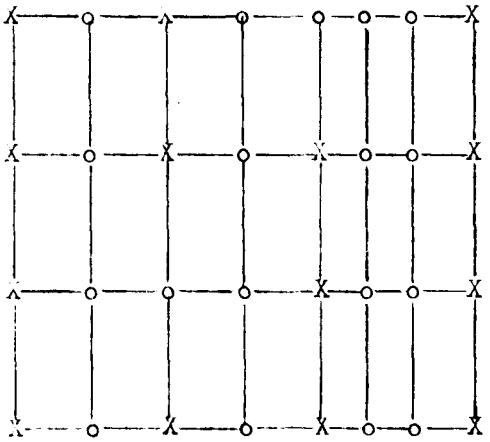


Fig. 5.4 An example finite element pattern for the column layout in Fig. 5.3.

o element mesh node X column coincident mesh node.

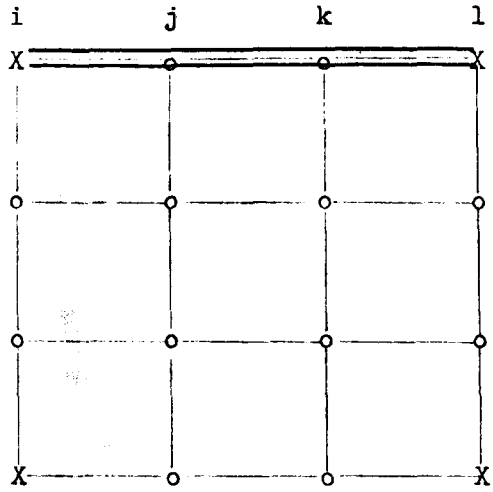


Fig. 5.5 Beam member and its corresponding beam elements.

o finite element mesh node.

X column coincident mesh node.

═══ beam member.

any flexibility in the use of finite elements over the deck area. For example it could not handle the column placings given on the deck area illustrated in Fig. 5.3.

Therefore, in order to remove the restrictions, it was decided to allow all the floor elements to be defined by the user with one restriction - that all columns on the floor must be coincident with nodes of the finite element floor mesh. Then it would be possible to fit a finite element pattern to the floor area shown in Fig 5.3 and one example is given in Fig. 5.4.

The provision for a choice of finite elements now allows for the inclusion of a finer mesh where greater variation of results is expected. Fig. 5.4 gives an illustration of a graded mesh pattern.

Once the method of defining the mesh pattern for the floor was decided, the problem of accounting for compatibility at the slab/beam interface was easily overcome. It was done by incorporating the beam matrices into the floor matrix and maintaining compatibility at mesh points coincident with the beam between column supports i.e. a beam member between column supports i,l can be assembled into the floor matrix as beam elements i-j, j-k and k-l, as illustrated in Fig. 5.5.

The third problem, that of the in-plane rotational stiffness of a slab, is not so much a computational problem as a theoretical one. It will be apparent from section 4 that the plate elements described do not refer to this variable. At present there is no well tried way available for the incorporation of this parameter.

A plane stress finite element analysis using this extra

parameter, i.e. θ_z, at nodal points may lead to a possible solution but as yet no formulation of a suitable shape function has been forthcoming. Another possible approach could be the instigation of a deep beam analogy; however, it is more usual to overcome the problem by a computational technique, where the plate appears perfectly rigid with respect to this in-plane variable. Here usually torsional effects on a structure become invalid due to the fact that the rigid plate does not transmit torsional forces to other elements.

For this program an investigation into these possible theoretical solutions for the rotational problem were examined but none proved satisfactory and therefore, in order to maintain compatibility between structural elements, a direct stiffness coefficient system was produced. This has the effect of making the plate perfectly rigid to in-plane rotation but does allow for any rigid-body torsional effects to be transmitted to the columns.

5.4 Floor assembly.

In the DECK1 program each floor is assembled independently, and for each one two types of stiffness matrix are set up. The first is a flexural matrix and the other a plane stress matrix - both constructed with the use of finite elements as reported in section 4. These floor matrices are formed in preparation for a 'condensation' process which will create a reduced matrix relating variables that only refer to floor/column coincident nodes.

In a simple finite element slab analysis a sequential numbering system for the mesh permits an optimum bandwidth solution of the resulting matrix. Unfortunately when condensing a matrix, where sequential mesh numbering has been adopted, the sequence is such that usually the optimum bandwidth is lost because nodes that were previously connected to each other via other nodes are now directly connected. The form of the matrix required for condensation is as follows:-

for the usual stiffness relationship

$$K \cdot d = P \tag{5.1}$$

where K is the stiffness matrix,

d is the displacement vector

and P is the load vector.

Then if variables r_1 are to be removed from d by condensation and r_2 retained then the equation (5.1) is required in the form of

$$\begin{vmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{vmatrix} \begin{vmatrix} r_1 \\ r_2 \end{vmatrix} = \begin{vmatrix} p_1 \\ p_2 \end{vmatrix} \tag{5.2}$$

For further discussion on the numerical process of condensation the reader is referred to the 'Tee-beam' report of section 2.

Examples for the numbering of floor meshes with interacting columns will show that the retained nodes will occur randomly and can only be made to appear in the appropriate positions by an improvised numbering scheme. However, when this is done, the optimum bandwidth, which was so valuable with respect to storage, is lost. Yet, if condensation is not employed, it would be necessary to add the complete floor matrix into the

structure matrix. Thus there are advantages and disadvantages in this approach but with respect to this problem as a whole the gains are greater.

The floor matrix is held in a one-dimensional array which corresponds to the upper triangular part of the stiffness matrix for the floor. It is assembled element by element with column-interacting nodes located in the correct positions for condensation. These positions are obtained via the specification of the interacting nodes as such at the input stage. The matrix, when complete, is condensed and the resulting matrix and load vector are stored on disc. All floors are treated similarly before the program moves on to the next stage. Floors are classed as 'types' and only different types are formed so that identical floors need not be handled more than once. The condensation is applied to both the flexural and plane stress cases where the only difference between the two is the finite element used.

The loading for each floor is confined to a uniformly distributed load for the flexural case and point loading at mesh nodes for the in-plane stress case. Further expansion to cover greater variety in loading would not prove too difficult. A flowchart for the floor assembly of a particular 'type' up to the point of storing the condensed matrices is given in Fig.5.6.

5.5 Supporting-beam assembly.

The inclusion of beams into the analysis produced the need for continuity between slab/beam interface. Ordinarily the beams

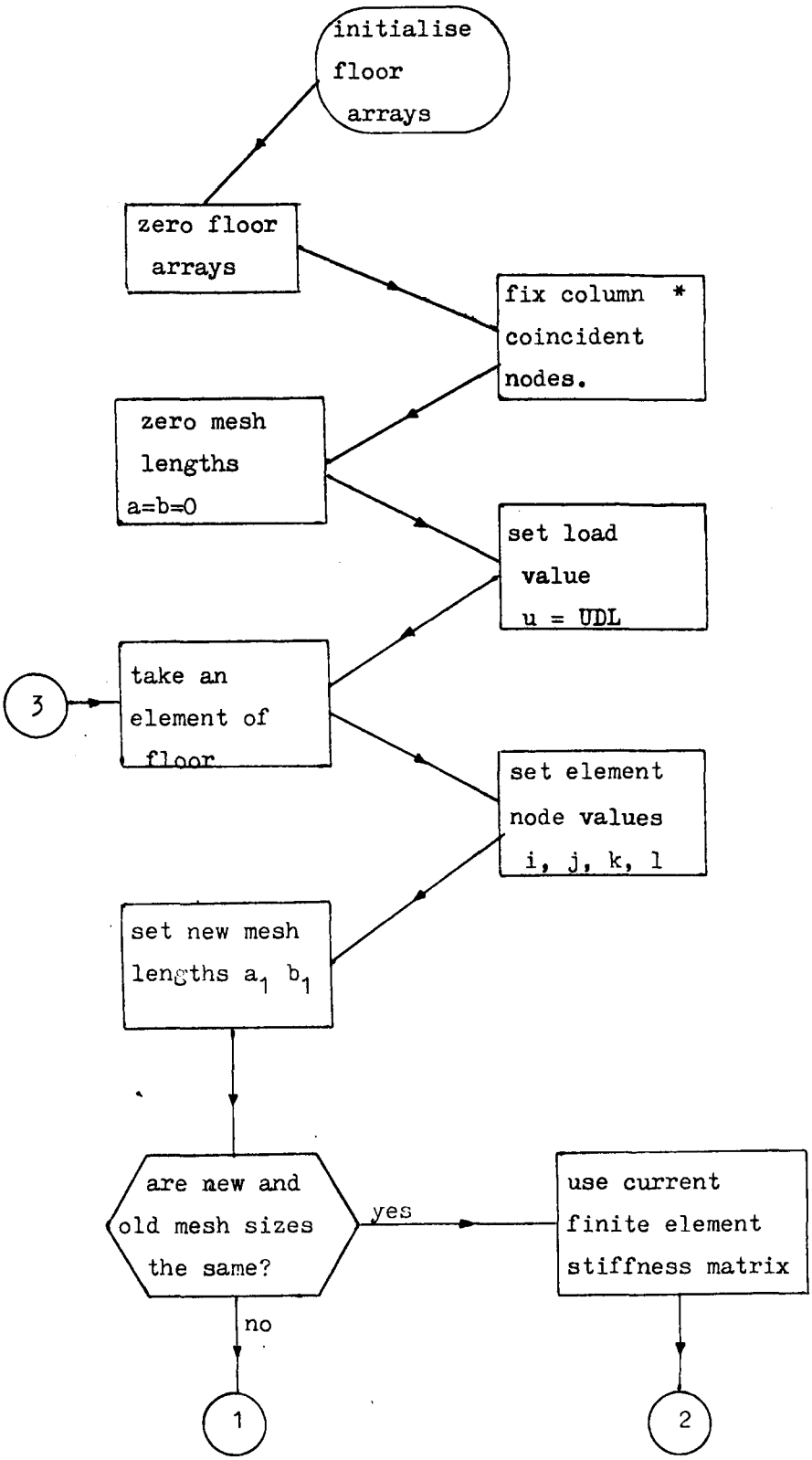


Fig. 5.6 Flowchart for floor assembly.

(Used for both flexural and shear assembly.)

* via procedure ORDER.

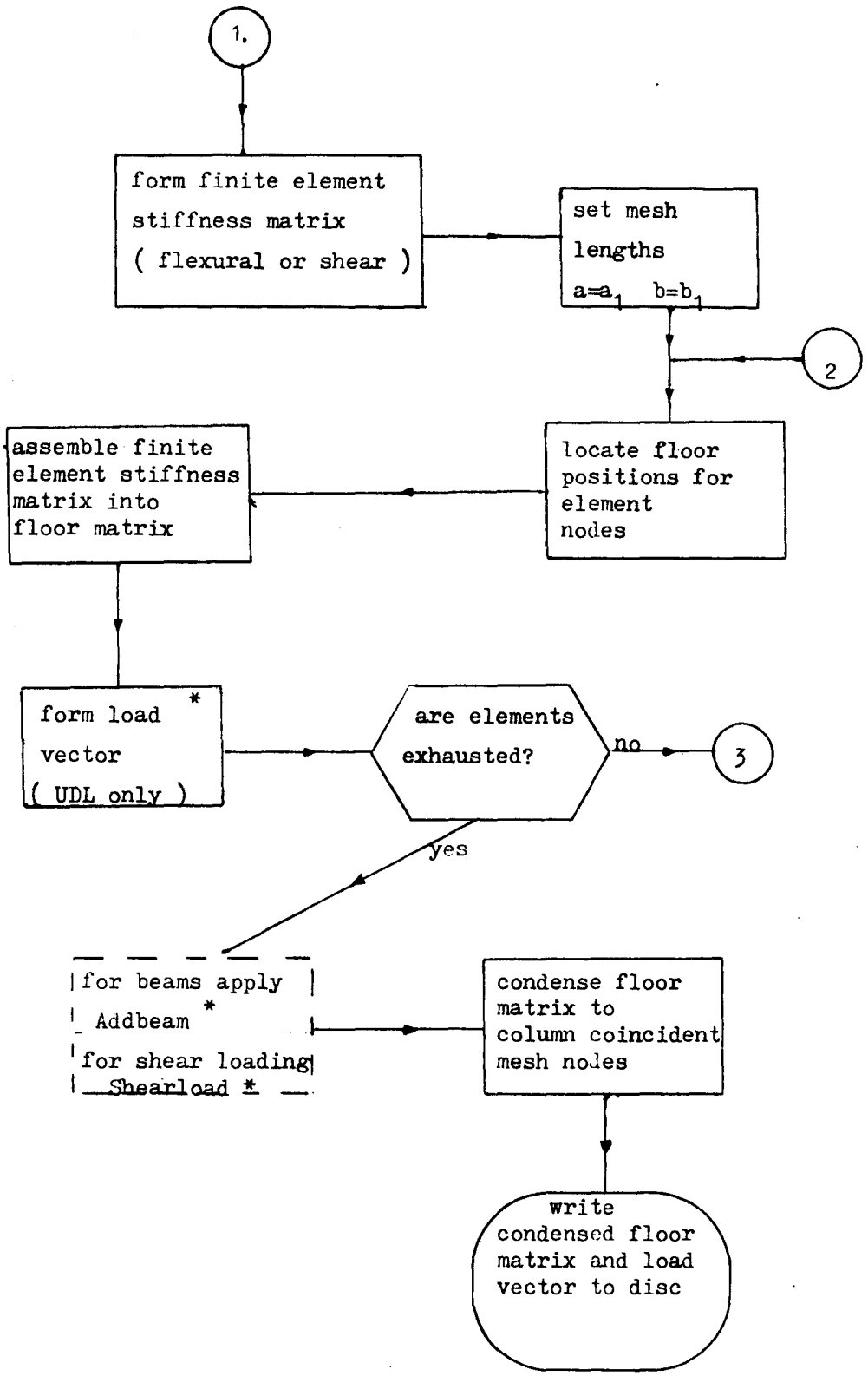


Fig. 5.6 cont. Flowchart for floor assembly.

(Used for both flexural and shear assembly.)

* These procedures are used whenever applicable.

would be handled in a similar fashion as the column members as in the skeletal frame program and assembled directly into the structure stiffness matrix. However, if this is done here continuity between slab and beam will only be maintained at beam ends. Therefore, since the floors are assembled independently of the main structure matrix, it was decided that supporting-beams should be incorporated into the floor matrices.

As two floor modes (flexural and shear) are used there is a need for the two compatible supporting-beam matrices. The supporting-beams are input as members identical to that for columns and are located via structure nodes e.g. beam i - j runs between structure nodes i and j. The program determines the beam members, locates the floor mesh points corresponding to the member ends and then calculates all the interspacing mesh points along the length of the beam. It then forms a stiffness matrix of the required type for each element length of the beam run and assembles it into the floor matrix, beam matrices being constructed using the factored form[1,2].

The beam assembly is computed via one procedure called 'ADDBEAM' which, for a given floor and floor mode, assembles all beam effects on that floor and of that mode into the appropriate matrix. This procedure is positioned in the flow-chart for the floor assembly given in Fig. 5.6. A flowchart for the Addbeam procedure is shown in Fig. 5.7. It has been designed as a multi-purpose routine and its further uses will be explained later in this section as they occur.

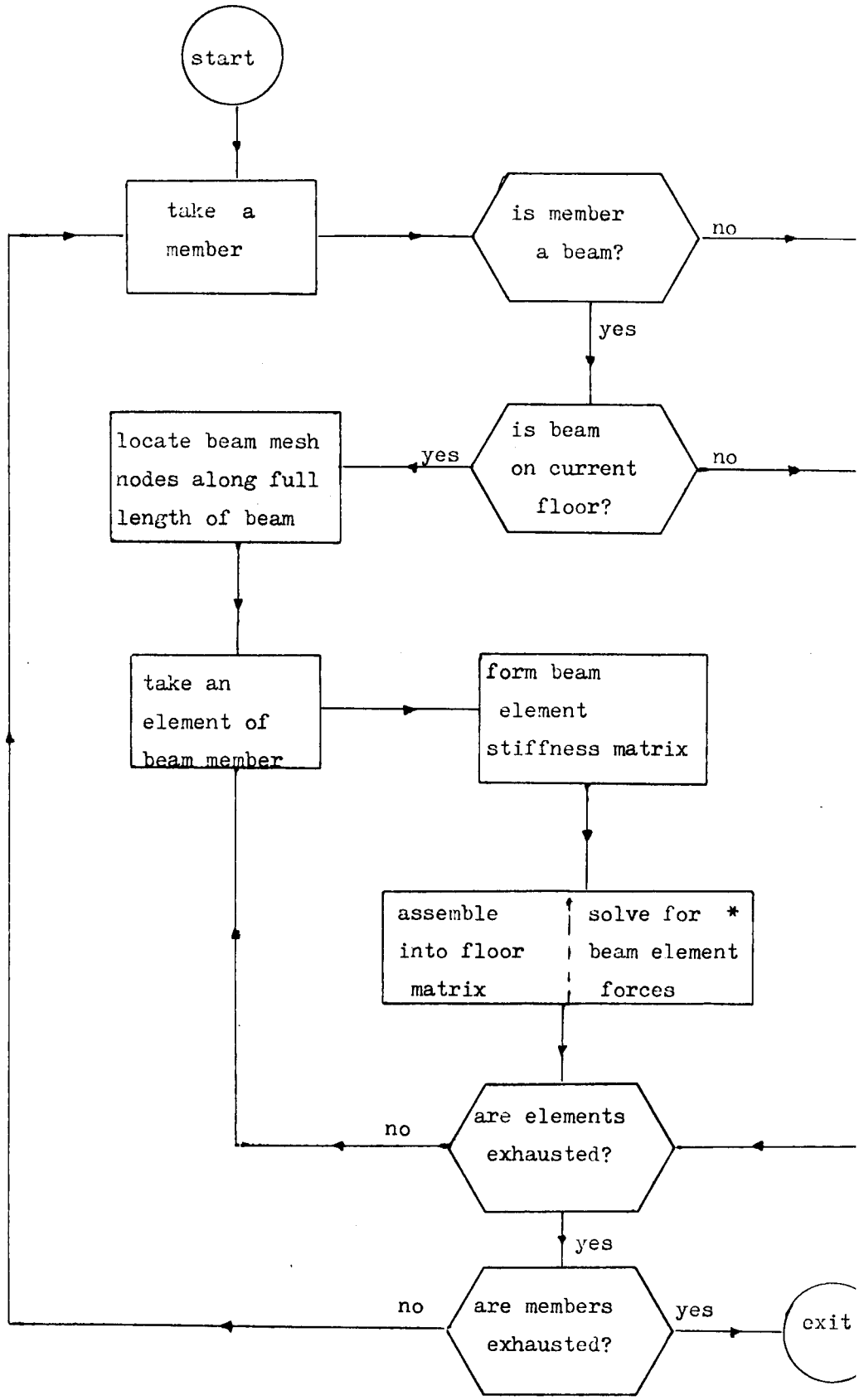


Fig. 5.7 Flowchart for ALDBEAM procedure .

* Alternative operation used for solution of beam forces.

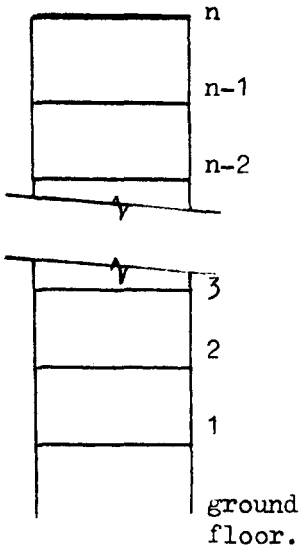
5.6 Structure assembly.

The obvious approach at this stage would be to construct the structure stiffness matrix based on the column nodes; however, here further use of the variable elimination technique was undertaken. It was decided to construct what can best be described as a 'part-structure' matrix. It is in size equivalent to the matrix for structure nodes for two floors only. The difference is that one part is a condensed matrix for the structure below the current level, related to nodes one floor below this current level. The other part is that section of the structure stiffness matrix related to the current level.

Thus the part-structure matrix is capable of holding two current floors and is changed progressively floor by floor until the top level is reached. This process is shown diagrammatically in Fig. 5.8. After condensation, matrices are written to disc, being held in readiness for a back-substitution process. This process travels in the opposite direction to the assembly i.e. it cascades down through the floors.

It is evident that in order to reduce working space the storage space required has been increased. However, this increase is not substantial, with the advantage that no operations are performed on matrices whilst they are in store. Compared to methods using one large, stored, stiffness matrix, where transfers from store to working space are great, this program's approach proves more efficient.

The structure assembly process is straightforward once all the floor matrices and vectors have been stored on disc in



typical structure.

1st. stage -

Initialise 'part-structure matrix'.

$$\left|\begin{array}{c|c} \text{ground} & \\ \hline & \text{1st} \end{array}\right| \left|\begin{array}{c} r_g \\ \hline r_1 \end{array}\right| = \left|\begin{array}{c} R_g \\ \hline R_1 \end{array}\right|$$

Now after condensation and adjustment the second floor is added.

2nd. stage -

$$\left|\begin{array}{c|c} g+1 & \\ \hline & 2\text{nd} \end{array}\right| \left|\begin{array}{c} r_1' \\ \hline r_2 \end{array}\right| = \left|\begin{array}{c} R_1' \\ \hline R_2 \end{array}\right|$$

Therefore generally equations are of the form -

i+1 stage -

$$\left|\begin{array}{c|c} g+1..+i & \\ \hline & i+1 \end{array}\right| \left|\begin{array}{c} r_i' \\ \hline r_{i+1} \end{array}\right| = \left|\begin{array}{c} R_i' \\ \hline R_{i+1} \end{array}\right|$$

until the n^{th} floor is reached-

n^{th} stage -

$$\left|\begin{array}{c|c} g...+n-1 & \\ \hline & n^{th} \end{array}\right| \left|\begin{array}{c} r_{n-1}' \\ \hline r_n \end{array}\right| = \left|\begin{array}{c} R_{n-1}' \\ \hline R_n \end{array}\right|$$

Fig. 5.8 Illustration of 'Part-Structure Matrix' Assembly.

their condensed form. The flowchart given in Fig. 5.10 shows the loop process undertaken for each floor. The cycle consists of assembling the current floor matrices, both flexural and shear types, and their interconnecting column stiffnesses from the floor below. It also accounts for any prescribed nodal restraints on that floor and fills the load vector where appropriate. Next condensation and storage stages are required, following which the 'part-structure' matrix is re-structured in preparation for the next floor cycle. The re-structuring is explained diagramatically in Fig. 5.9.

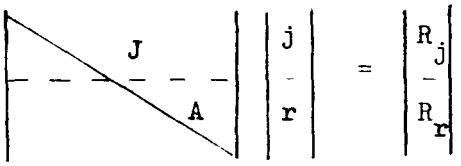
The assimilation of the condensed floor matrices, stored on disc in readiness, into the 'part-structure' matrix is undertaken via the implementation of a procedure called FLASSEM which locates the positions in the matrix for both the flexural and shear cases. The program uses internal procedures for the transfer of matrices to and from disc.

Column members for the lift between the floor levels under construction are assembled via the procedure ADDCOL which generates member stiffness matrices for the columns and places them into the 'part-structure' matrix where required.

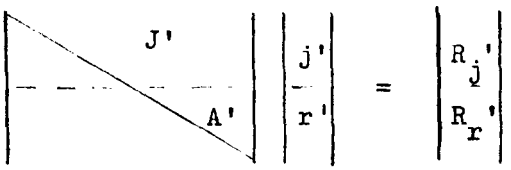
The condensation is performed by a general elimination procedure used throughout the program. It condenses the 'part-structure' matrix to an internally determined depth, so that the current sub-structure stiffness is concentrated on only the current floor's nodal variables. After condensation, the 'part-structure' matrix is 'adjusted' and the re-cycle begins. The re-cycling is stopped after the assembly of the last floor and before any condensation is implemented. At this

Fig. 5.9 Diagrams showing matrix structure adjustments necessary for the re-cycling process.

Before the condensation stage in the structure assembly the equations concerning the 'part-structure matrix' have the form as follows:-

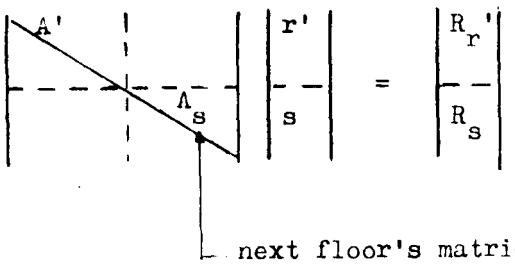


the 'part-structure matrix' is then condensed arriving at



Now for the recycle the condensed matrix A' has to be placed in a new position so that the next floor can be accommodated.

i.e. equations take the following form:-



Note.

In the structure assembly each floor is assumed to occupy half the size of the 'part-structure matrix' i.e. A' is the same size as A_s. However this is not always true e.g. where a change in floor pattern occurs. In order to account for this occurrence a procedure MATSET is instigated to take account of this difference in size.

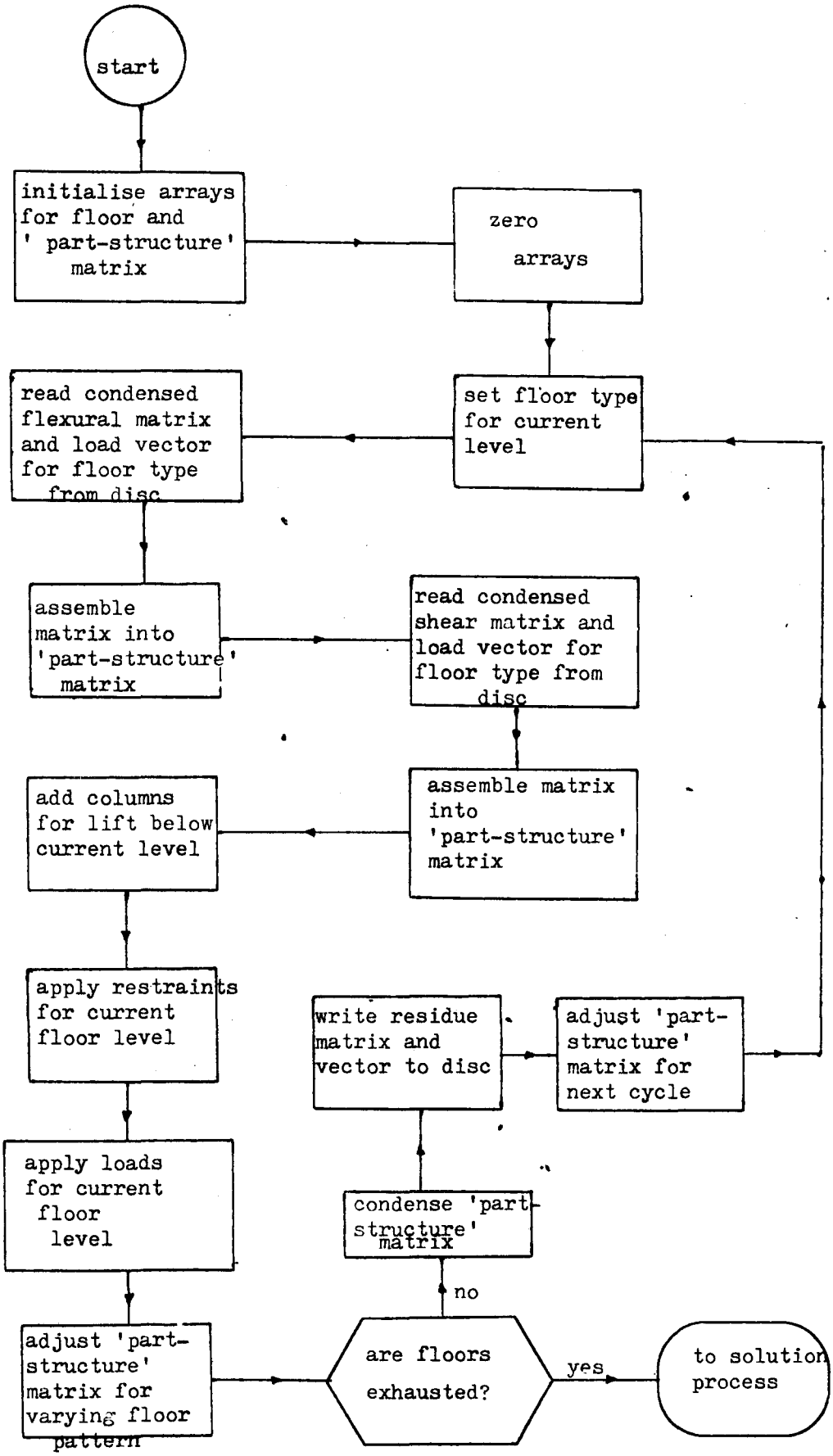


Fig. 5.10 Flowchart for structure assembly.

stage the solution for structure displacements begins.

5.7 Solution for structure deformations.

The solution for the total structure displacements is undertaken in two parts. Initially the solution for the deformations of the structure nodes is determined via the back-substitution process mentioned in the previous subsection. When the final floor has been assembled into the 'part-structure' matrix, instead of the condensation, a full nodal displacement solution for the top two levels is obtained. At this point the stored matrices are retrieved from disc and having the solved displacements a back-substitution will enable the calculation of the next lower floor's displacements. The back-substitution can be shown algebraically as follows:-

for a 'part-structure' system $K.d = P$

or

$$\begin{vmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{vmatrix} \begin{vmatrix} d_1 \\ d_2 \end{vmatrix} = \begin{vmatrix} P_1 \\ P_2 \end{vmatrix} \qquad (5.3)$$

cf (5.1)

Now d_2 is known and the matrix stored on disc corresponding to $\begin{vmatrix} k_{11} & k_{12} \end{vmatrix}$ gives an equation of the form:-

$$\begin{vmatrix} \ddots & \underline{A} \\ \underline{0} & \underline{B} \end{vmatrix} \begin{vmatrix} d_1 \\ d_2 \end{vmatrix} = \begin{vmatrix} P_1 \\ P_2 \end{vmatrix} \qquad (5.4)$$

where $\underline{0}$ is Null Matrix.

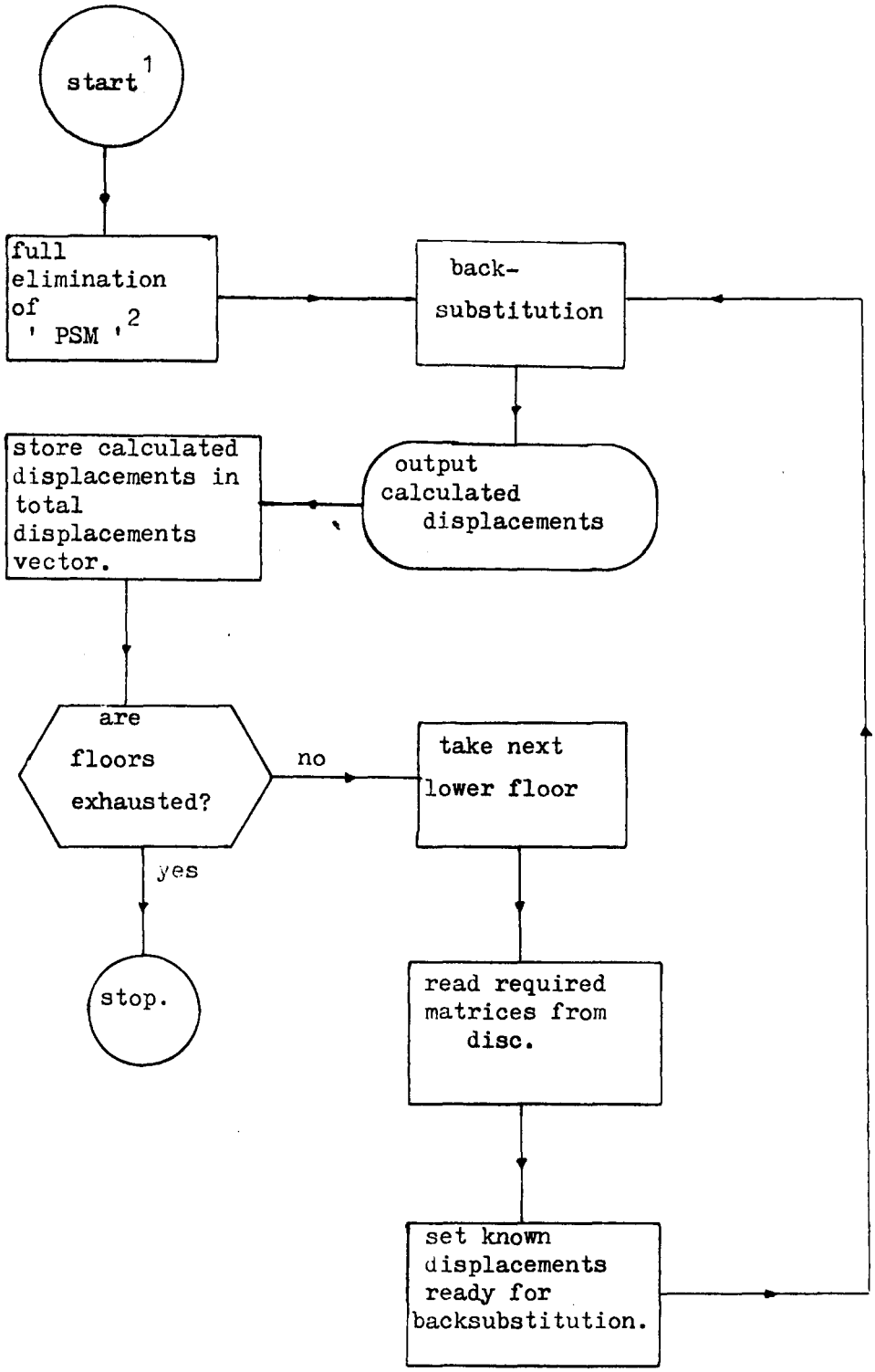


Fig. 9.11 Flowchart for solution of structure displacements.

1 From assembly section. 2 'Part-structure matrix'.

This can now be solved for d_1 by a simple back-substitution. Thus taking each matrix in turn from disc it is possible to travel down the structure solving for displacements floor by floor, and the flowchart for this part of the DECK1 program is given in Fig. 5.11.

Internal floor displacements.

The internal floor displacements (i.e. those displacements associated with mesh points not coincident with structure nodes) are derived by re-constituting the appropriate floor matrix and performing a back-substitution utilizing known displacements at the points of column interaction. The flowchart for this segment is similar to that for the floor assembly (Fig. 5.6) but with certain changes as follows:-

(i) The insertion of known displacements - via procedure RESIN2.

(ii) A full solution for the floor i.e. a complete elimination followed by a back-substitution.

(iii) The output of floor displacements - via procedure DISPLAY2 - replacing the writing to disc stage.

As with the floor assembly, this section is undertaken in two parts - i.e. a flexural and plane stress case. Known mesh displacements, obtained from the structure solution, are added to the floor displacement vector allowing the condensation to be taken to a full solution.

5.8 Solution for member and element forces.

The force components calculated by the program comprise of the member end forces for columns and beam elements and the internal bending moments of each floor. It does not, as yet, produce the stresses associated with the in-plane loading on a floor.

The member end forces for columns are calculated in a similar fashion to that described in section 3 - 'Orthogonal Space Frame Program'. Since the supporting beam effects are built into the floor matrix from mesh point to mesh point, the force output is given for each element of a supporting beam. The procedure 'Addbeam' performs all the operations necessary to calculate these forces. As explained earlier in this section this is a multi-purpose procedure, which, besides handling the supporting beam assembly, also handles the production of supporting beam forces. The flowchart is similar to that given in Fig. 5.7. The only difference is that instead of constructing the floor matrix, each element stiffness matrix is used to calculate element forces. The procedure is used twice, once to obtain flexural forces and then again for shear forces.

The method of calculation of floor element forces is that described in section 4. The forces determined are the usual bending and twisting moments per unit length related to the element axes i.e. M_x, M_y, M_{xy} - the usual notation for these forces. The principal moments are easily deduced from these basic moments. For a full explanation and discussion of this topic and its reference to the design of reinforcement

the reader is referred to the User Manual and Report of the Ministry of Transport - 'MOT/CIRIA Finite Element Package for the Analysis of Reinforced Concrete Slab Bridge Decks[3]'.

The floor element and supporting beam forces are determined as each floor's displacements become available. However, the column member-end forces are not calculated until the program has completed the solution for all floors and they occur as the concluding operations of the program.

5.9 Systems Usage.

With a program of this nature, it is apparent that a certain amount of dependence on the systems used in development is inevitable. It would be advantageous to be able to run the program on various machines but in reality this is usually impracticable due to this built-in dependence on the original system used.

The program, DECK1, although using general principles for its solution of the structural problem, does have these built-in deficiencies. However, instead of suffering this hindrance passively, it was decided to make the best use of what was at hand.

The program was developed on an ICL Elliot 4130 machine which was operated on a batch process system. The total core store of the computer was 96k, but for general use it was usual for the maximum available to be 64k. Backing store was available in the form of either disc or magnetic tape and under systems usage disc workfiles were also available. There were

five files in all giving a total of 175k, without the need of initialising an independent tape or disc, since the workfiles existed on the systems disc.

For the purposes of the DECK1 program it was decided not to create backing store from new disc or tape units but to utilize the workspace on the systems disc. The store available is quite considerable but does fall a long way short of that which could have been obtained from other sources. Tape or disc backing store is usually best utilised for data storage between program usage and not as in this case used as data storage during run time. The working arrays are kept to a useable size in keeping with the available core store and by utilizing workfiles data transfers are kept to a minimum.

It is apparent that the built-in deficiencies of this program revolve around the backing store system employed and to make the program compatible for other machines it is this process which would need to be adapted. The program is written in '4100 ALGOL' which is almost the same as ALGOL 60 but with the modifications of the 4100 system.

5.10 Program Implementatation.

The program will only handle one set of loading data per run due to the complexity of operations which the loading vectors undergo. All loadings are applied nodally and are restricted to the following:-

(1) Structure loading.

Point load or moment applied in any of the structure axes directions - M_x, M_y, M_z, P_x, P_y, P_z.

(2) Floor loading.

Uniformly distributed vertical load - U^*.

In-plane point loads in element axes directions - F_x, F_y.

* Converted internally to nodal loading at mesh points on the floor deck.

No capacity for moments to be applied on the floor area has yet been included.

The loading system has purposely been kept simple, designed only to provide a system which allows easy use of the program. A sophisticated loading system with a full syntax check, as was used for the Tee-beam program, which was described in section 2, could be added.

All loads applied to the structure, as in (1) above, are processed by a procedure called ADDLOAD which handles these loadings floor by floor as they are required. The other loadings are more complex. Instead of the input of loads for all floors, it is only necessary to supply loads for different 'types' of floor. (The term TYPE is as defined earlier in this section.) Therefore the UDL and shear loads for the floors are input for the number of types rather than the number of floors. The uniformly distributed loading is processed internally by the floor assembly section. Due to the regeneration of floor matrices later in the program it is necessary to store both the UDL and shear loadings. The UDL is easy to store as there is only one

parameter per floor type. However, the shear loads are more complex and are handled by a procedure SHEARLOAD which applies the loads when required.

Mode of Nodal Restraint.

The basic method of restraining variables is used here i.e. making one node appear fixed by adjustment to its direct stiffness coefficient. It is possible to restrain any of the structure nodes in this way. Each node can be restrained from movement for each degree of freedom. The whole process is handled by ADDREST which stores the restraints as input and enforces them as required. There is a possible maximum of 6 restraints per node - 3 deflections and 3 rotations.

An addition to the program has made it possible to restrain a floor deck not only at structure nodes, but also at any floor mesh node. Therefore a built-in edge could be approximated by restraining all the necessary edge mesh nodes. The parameter MAXLINK specifies the maximum number of restraints on any floor and each floor's quota is supplied later in the input. A procedure called FLOFIX implements the floor restraints for both the flexural and shear modes.

The program has not been written to operate in any specified units - thus allowing choice of units to the user. However, certain drawbacks are inevitable when this is done. It is necessary that the user must be consistent in the use of units throughout the input and expect the output to be in compatible units. Also it is necessary to be wise in the

choice of such units because the specified output format may not be capable of printing results in the desired form.

Units are required for the input of quantities such as force, length etc. and the following are some suitable input unit systems which can be used for this purpose:-

lb.f. --- inch.

KN. --- metre.

KN. --- mms.

5.11 Program and Flowcharts.

Flowcharts for certain procedures and the more important parts of the program body have been given in the text where they occur. Flowcharts for the remaining portions will not be given; however, a full Algol listing of the DECK1 program is given in the Appendix.

5.12 Note on Sign Convention.

The sign convention used for joint displacements and member end forces is the same as that given in section 3. This system is clockwise positive for moments, and is incompatible with a sign convention based on sagging and hogging moments. If the reader is in any doubt of this he is referred to Livesley's explanation

of this situation[1].

It is important to note that for a given member e.g. 1 - 6, joint 1 is end 1 and joint 6 is end 2. Therefore results for member end forces are given thus:-

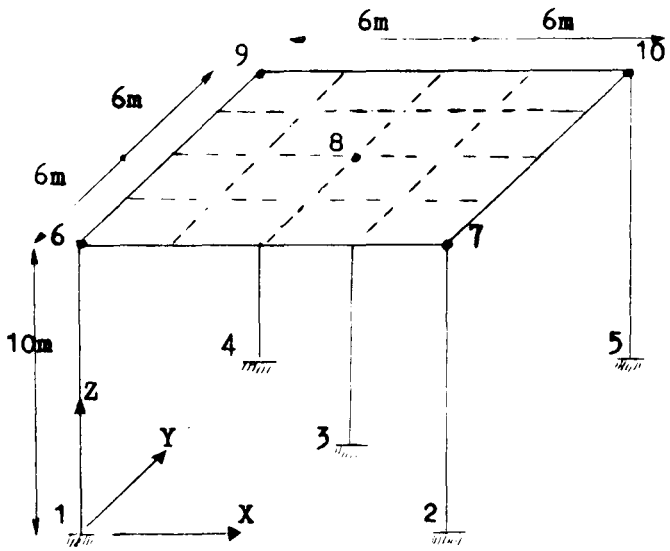
MEMBER	AX1	MY1	MZ1	MY2	MZ2	SY1	SZ1
1 - 6	50.0	15.0	15.0	30.0	30.0	1.0	-1.0

5.13 Test Frames Illustrating the Use and Versatility of the DECK1 Program.

Four example structures are analysed using the DECK1 program where each example is used to illustrate a different facet in the use of the program. Comparison of the results with other verified values is given in the following section and therefore will not be presented here.

Since the amount of output can be quite considerable it was decided to present only those results best suited to illustrate each example's use of the program. The method of presentation also varies from example to example.

Example 1: A structure not conforming to the use of plane frames.



Sketch of Structure for Example 1.

All columns 200x200mms.

$E = 1.378 \times 10^7$ KN/m^2

$G = 5.512 \times 10^6$ KN/m^2

$\nu = 0.150$

Floor: thickness 200mm, split into elements as shown.

Two load cases were analysed.

(a) Uniformly distributed load of $1KN/m^2$ on the floor area.

(b) As (a) plus a point load of 5KN in the X direction at joint 6.

Results for load case (a).

1) Structure joint displacement

TABLE 5.1.

Joint	DEFZ	θx	θy
6	-3	-16	16
7	-3	-16	16
8	-15	0	0
9	-3	16	-16
10	-3	16	-16

deflection - metres x 10^4

rotation - radians x 10^4

2) Floor displacements

Figs.5.13 and 5.14 give the vertical deflection and rotation for specific sections through the slab. The effect of the centre column is clearly visible.

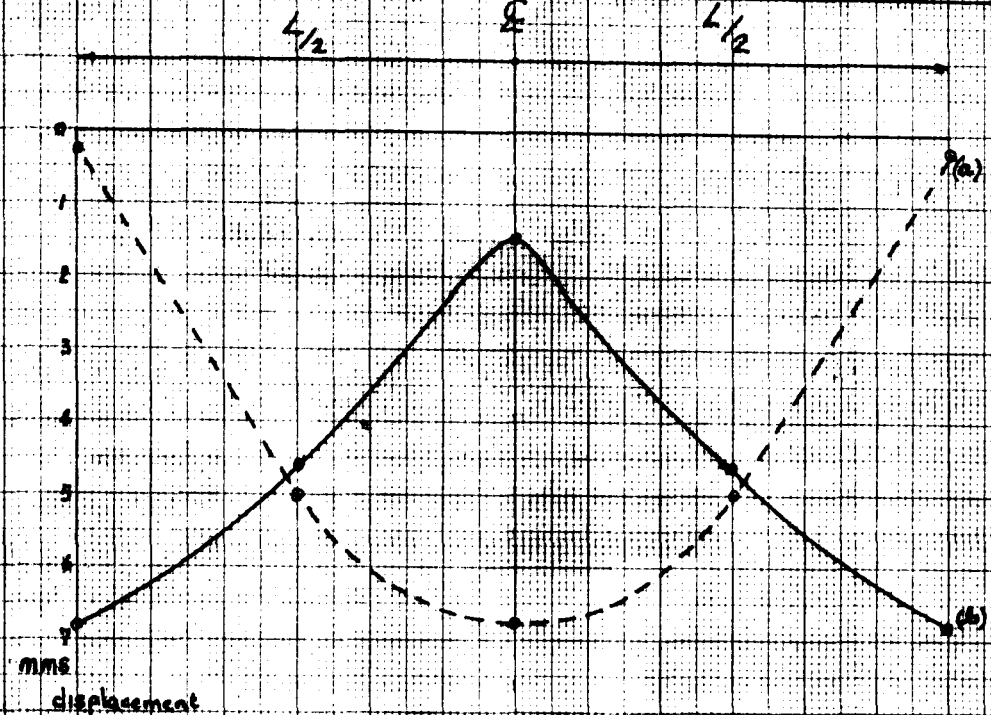
3) Column forces

TABLE 5.2.

Member	AX1	MY1	MZ1	MY2	MZ2	SY1	SZ1
1 - 6	15.0	0.6	0.6	1.2	1.2	0.2	-0.2
2 - 7	15.0	-0.6	0.6	-1.2	1.2	0.2	0.2
3 - 8	84.0	0.0	0.0	0.0	0.0	0.0	0.0
4 - 9	15.0	0.6	-0.6	1.2	-1.2	-0.2	-0.2
5 -10	15.0	-0.6	-0.6	-1.2	-1.2	-0.2	0.2

Units: KN or KNm

Fig. 5.13 Vertical displacement of slab (load case 1 example 1) (a) deflection along one edge, (b) deflection across centre line of slab.



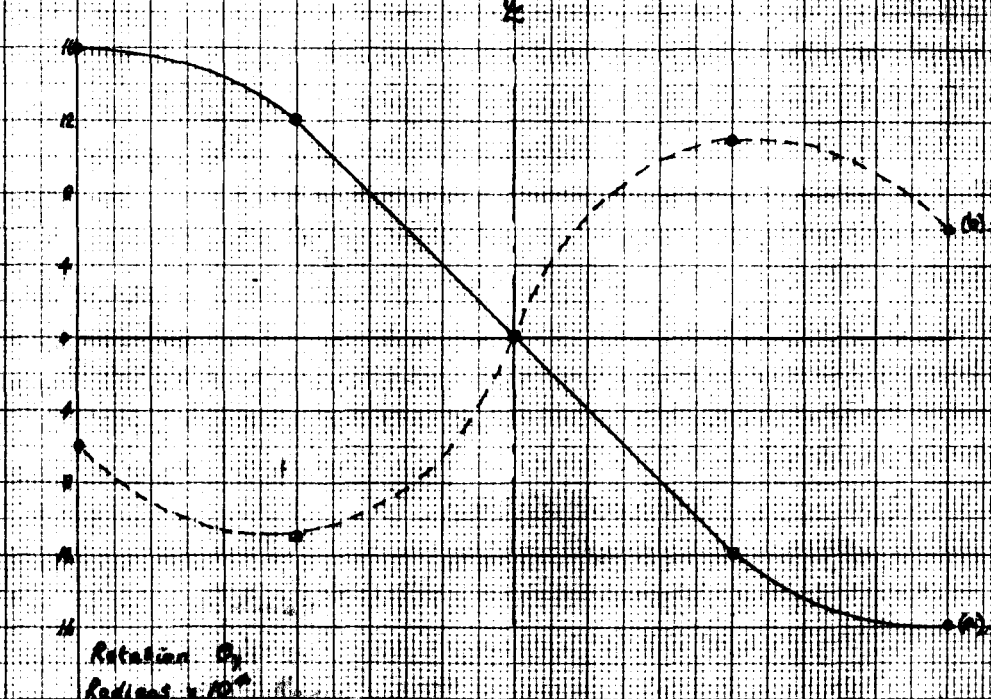
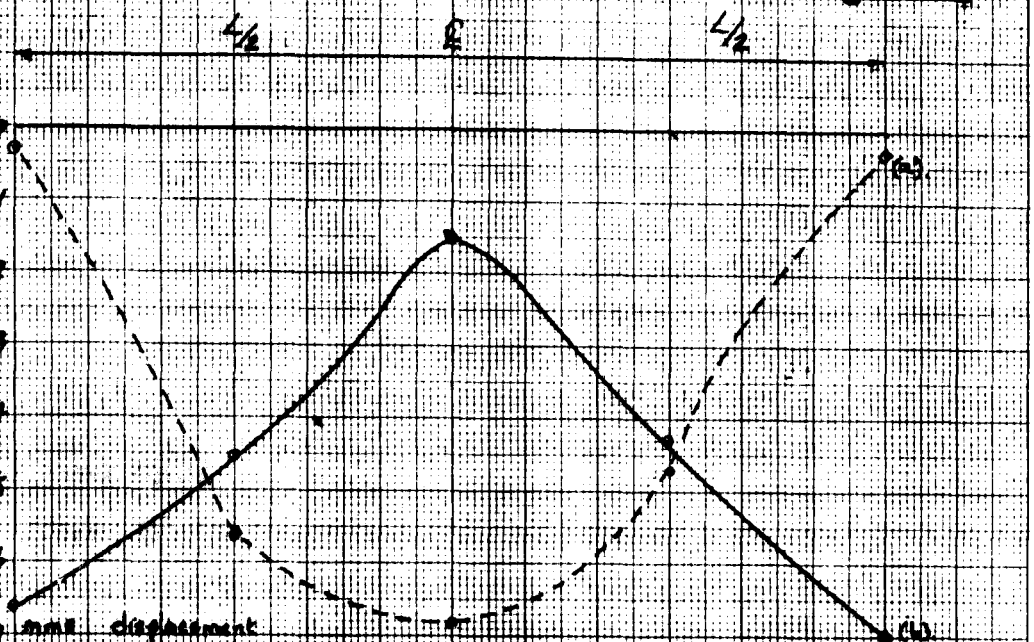


Fig. 5.14 θ_y Rotations along (a) edge, and (b) centre line of slab in y direction (load case 1 example 1).

Fig. 5.15 Vertical displacement of slab (load case 2 example 1) (a) along edge and (b) along centre line in x direction along slab.



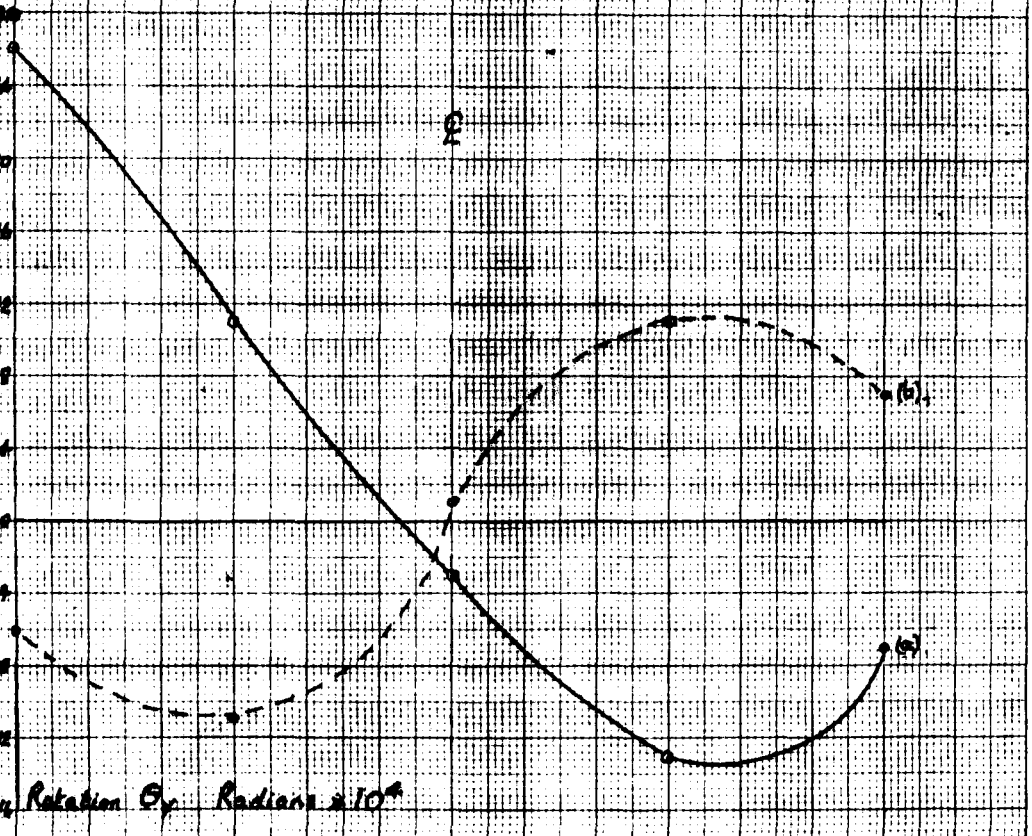


Fig. 5.16 θ_y Rotation along (a) edge, and (b) centre line of slab in x direction (load case 2 example 1).

Results for load case (b).

1) Structure joint displacements.

TABLE 5.3.

Joint	θx	θy	DEFX	DEFY	DEFZ
6	-12	26	782	-304	-3
7	-21	-7	782	304	-3
8	0	1	479	0	-15
9	19	18	175	-304	-3
10	13	-15	175	304	-3

deflection - m x 10^4

rotation - radians x 10^4

2) Floor displacements.

Figs. 5.15 and 5.16 give graphical representation of the vertical deflection and rotation for the same sections as in Figs. 5.13 and 5.14 for comparison with the results of load case (a).

3) Column forces.

TABLE 5.4.

Member	AX1	MY1	MZ1	MY2	MZ2	SY1	SZ1
1 - 6	14.0	-7.7	3.8	-6.7	4.2	0.8	1.4
2 - 7	16.0	-8.9	-2.6	-9.1	-1.8	-0.4	1.8
3 - 8	84.0	-5.2	0.0	-5.2	0.0	0.0	1.0
4 - 9	14.0	-1.3	2.6	-0.6	1.9	0.5	0.2
5 - 10	16.0	-2.5	-3.8	-3.0	-4.3	-0.8	-0.8

Units: KN or KNm

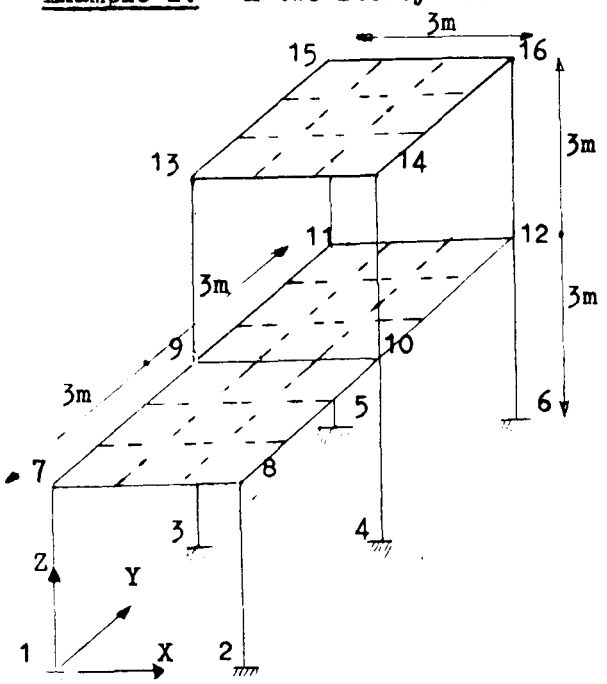
Comments on Example 1.

It should be noted that, even though it is of a highly symmetrical

nature, this structure could not easily or accurately be analysed by a series of plane frames. It is also important to note how the total floor load has been distributed between the columns. This is shown by both the vertical displacement and the corresponding axial load. The simple catchment area theory would provide, for a total load of 2P, P/4 at each corner column and P at the centre column. This corresponds to values of 18KN and 72KN for the loading in load case (a). However, the results give 15KN at each corner column and 84KN at the centre column. Thus a large discrepancy is inherent in the centre column load when using the simple area method for floor load distribution.

The structure displacements for load case (b) tend to support the assumption that the slab does not move as a rigid body, which had been assumed by some earlier workers in this field as stated in the Introduction.

Example 2: A two storey block with differing floors.



All columns 200x200mms

$E = 13.78\ KN/mm^2$

$G = 5.512\ KN/mm^2$

$\nu = 0.150$

Large floor thickness = 150mm

Small floor thickness = 120mm

The floors are split into elements as shown.

Sketch of structure for Example 2.

One load case, that of a uniformly distributed load of $1KN/m^2$ on the floor area, was analysed.

Results for Example 2.

1) Structure joint displacements.

TABLE 5.5.

Joint	ROTX	ROTY	ROTZ	DEFX	DEFY	DEFZ
7	-1	1	0	2	192	109
8	-1	-1	0	-2	192	109
9	0	1	0	-1	189	395
10	0	-1	0	1	189	395
11	1	1	0	-2	192	231
12	1	-1	0	2	192	231
13	-2	1	0	5	630	518
14	-2	-1	0	-5	630	518
15	2	2	0	3	634	353
16	2	-2	0	-3	634	353

deflections - mms x 10^4

rotations - radians x 10^4

2) Floor displacements.

Fig. 5.17 illustrates diagramatically the deformation of a 'part frame' of the three-dimensional structure. The effect in the reduction of the floor thickness with respect to the floor displacement is apparent.

3) Column forces.

TABLE 5.6.

Member	AX1	SY1	SZ1	MY1	MZ1	MY2	MZ2
1 - 7	2.00	0.13	-0.16	163.11	125.13	326.44	273.6
2 - 8	2.00	0.13	0.16	-163.11	125.13	-326.44	273.6
3 - 9	7.26	-0.05	-0.18	180.82	-60.11	361.51	-97.1
4 - 10	7.26	-0.05	0.18	-180.82	-60.11	-361.51	-97.1
5 - 11	4.24	-0.08	-0.07	75.05	-88.31	149.84	-153.2
6 - 12	4.24	-0.08	0.07	-75.05	-88.31	-149.84	-153.2
9 - 13	2.26	0.24	-0.35	528.75	233.25	516.16	477.6
10 - 14	2.26	0.24	0.35	-528.75	233.25	-516.16	477.6
11 - 15	2.24	-0.24	-0.28	350.48	-268.23	477.26	-442.6
12 - 16	2.24	-0.24	0.28	-350.48	-268.23	-477.26	-442.6

Units: KN or KNmm.

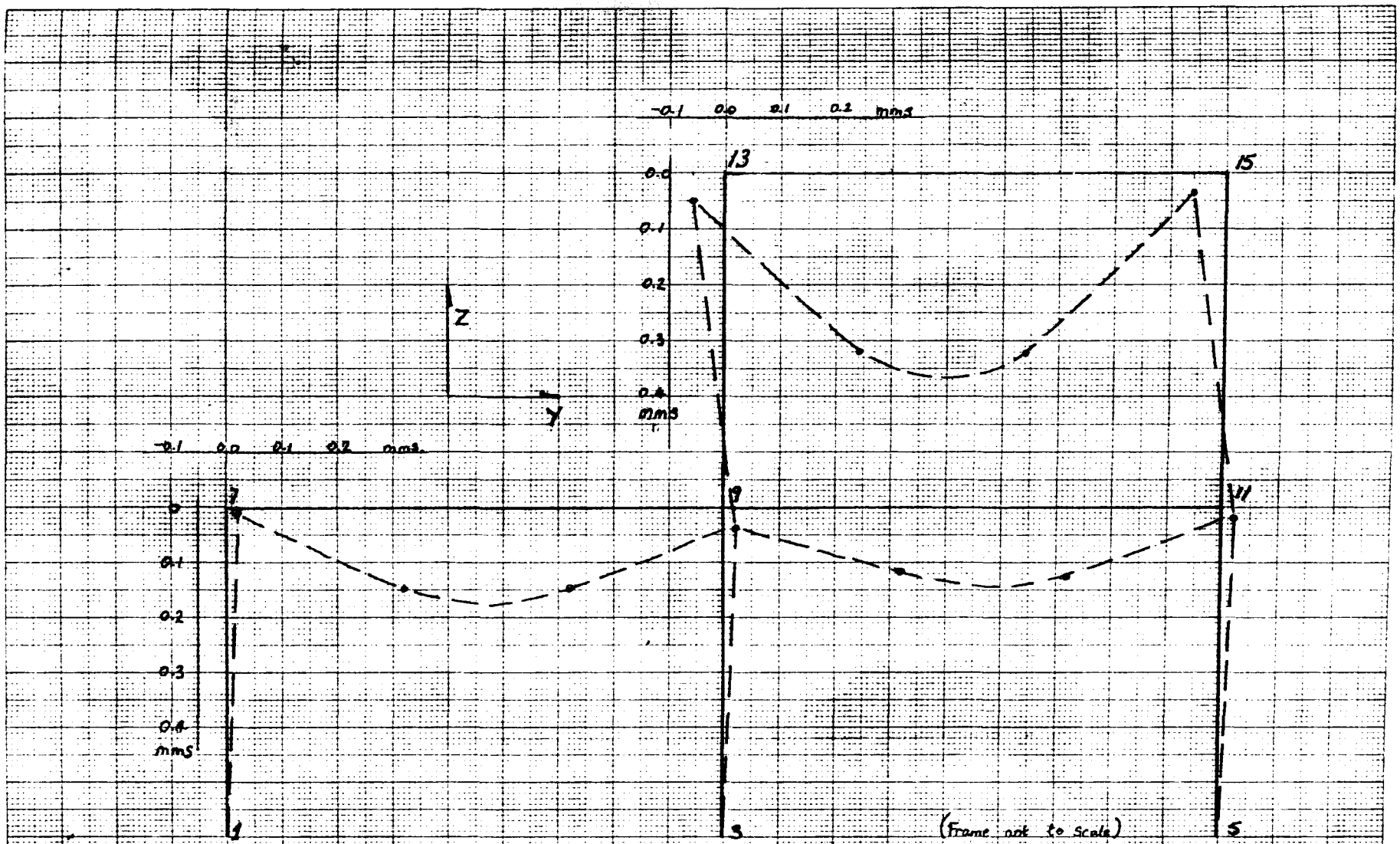
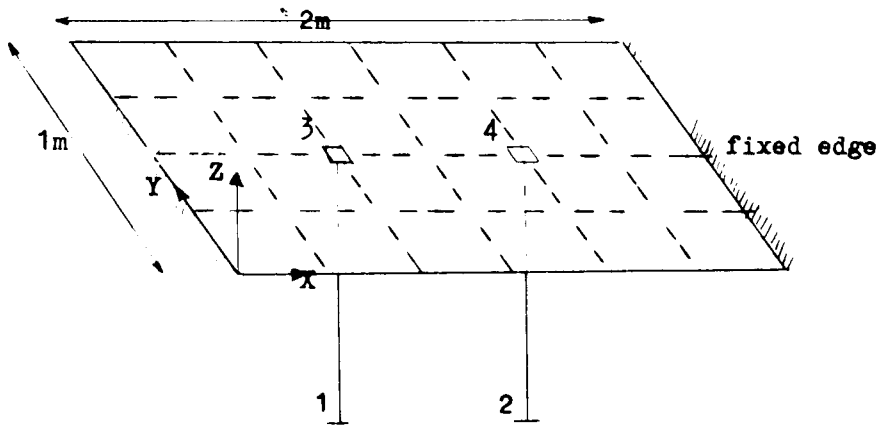


Fig. 5.17 Displacement of 'part frame' of example 2.
Deflection in Y and Z directions.

Comments on Example 2.

The example structure could have been analysed by the splitting of the space frame problem into a plane frame analysis. However, a two-dimensional analysis would not have shown the biaxial bending in the columns. If multi-framing is used the biaxial bending would become apparent but the actual moment values could only be very approximate. Also no information about the floor deformation would be forthcoming from a plane frame analysis.

Example 3: Model of a bridge deck.



Sketch of bridge structure for Example 3.

columns - 200x200mms

$E = 13.78\ KN/mm^2$

$G = 5.512\ KN/mm^2$

$\nu = 0.150$

floor thickness 120mm split into 24 elements as shown

A solution for the loading of $10KN/m^2$ uniformly distributed load on the deck area is given.

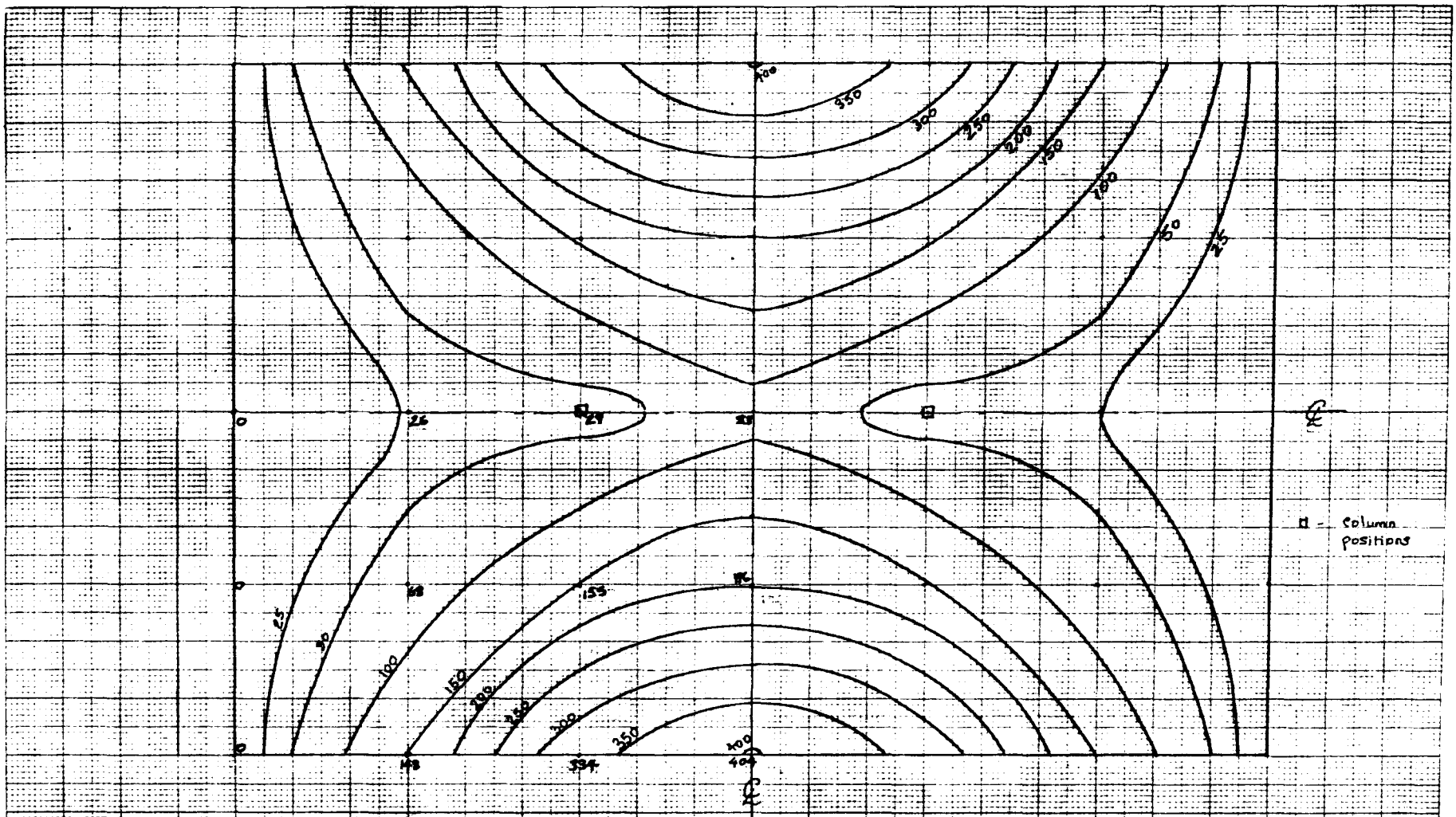


Fig. 5.18 Contour Plot of Vertical Deflection of slab deck

for example 3 (Displacements shown in mms $\times 10^3$)

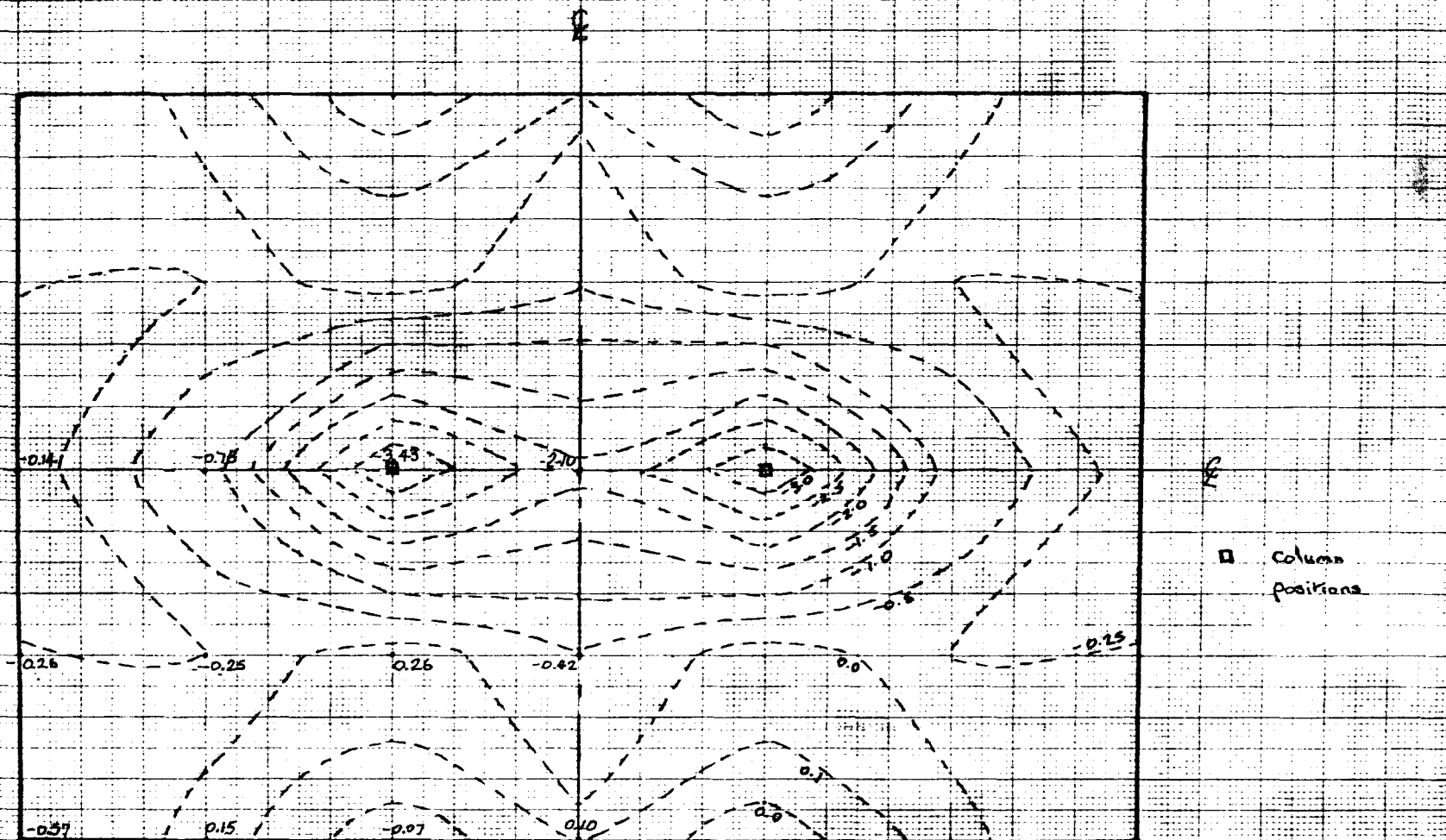


Fig. 5.19 Slab Moment M_y for Example 3 (Moment in kNmm/mm)

Results for Example 3.

1) Structure joint displacements.

TABLE 5.7.

Joint	DEFX	DEFZ
3	2	287
4	2	287

mms x 10^4

2) Floor displacements.

A contour plot of the vertical deflection over the deck is given in Fig. 5.18.

3) Floor moments.

A contour plot of the My moment per unit length for the deck is given in Fig. 5.19. A graph for Mx is given in the following section.

4) Column forces.

TABLE 5.8.

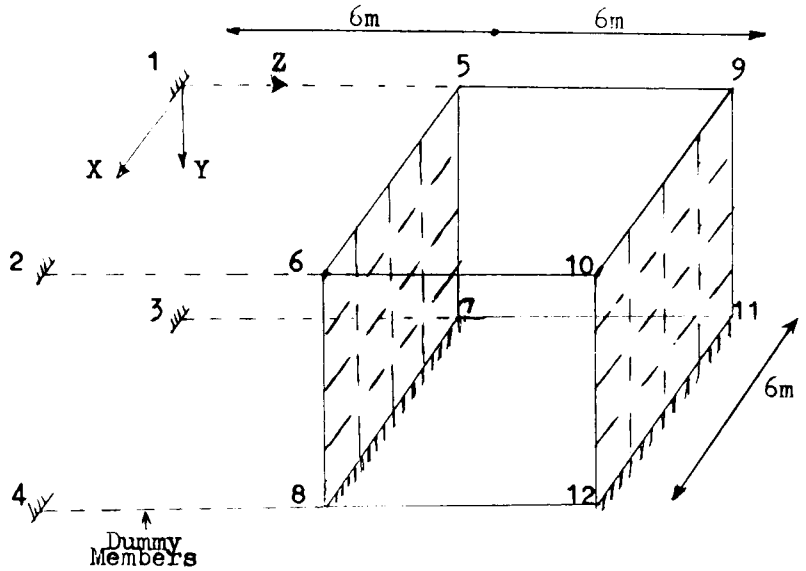
Member	AX1	SZ1	MY1	MY2
1 - 3	15.83	-0.51	169.65	341.38
2 - 4	15.83	0.51	-169.65	-341.38

Units: KN or KNmm

Comments on Example 3.

The example given illustrating the versatility of the program shows how a bridge deck could be analysed whilst accounting for the interaction between the deck and columns. The program also allows for the floor mesh points on the deck to be rigidly fixed.

Example 4: Analysis of a shear wall structure.



Sketch of structure for Example 4.

$E = 13.78\ KN/mm^2$

$G = 5.512\ KN/mm^2$

$\nu = 0.150$

wall thickness = 150mm, elements as shown.

The walls are edged by 200x200mm members i.e. 5 - 6, 5 - 7, 6 - 8. Other members are 200x200mm except for the dummy members, where small member stiffnesses have been used. These dummy members have been used to transform the structure into a form which the program can handle. However, due to their negligible stiffness no extra constraint is applied to the frame. One load case was analysed. The loading is illustrated in Fig. 5.20 where 5KN point loads were applied as shown.

Results for Example 4.

1) Structure joint displacements.

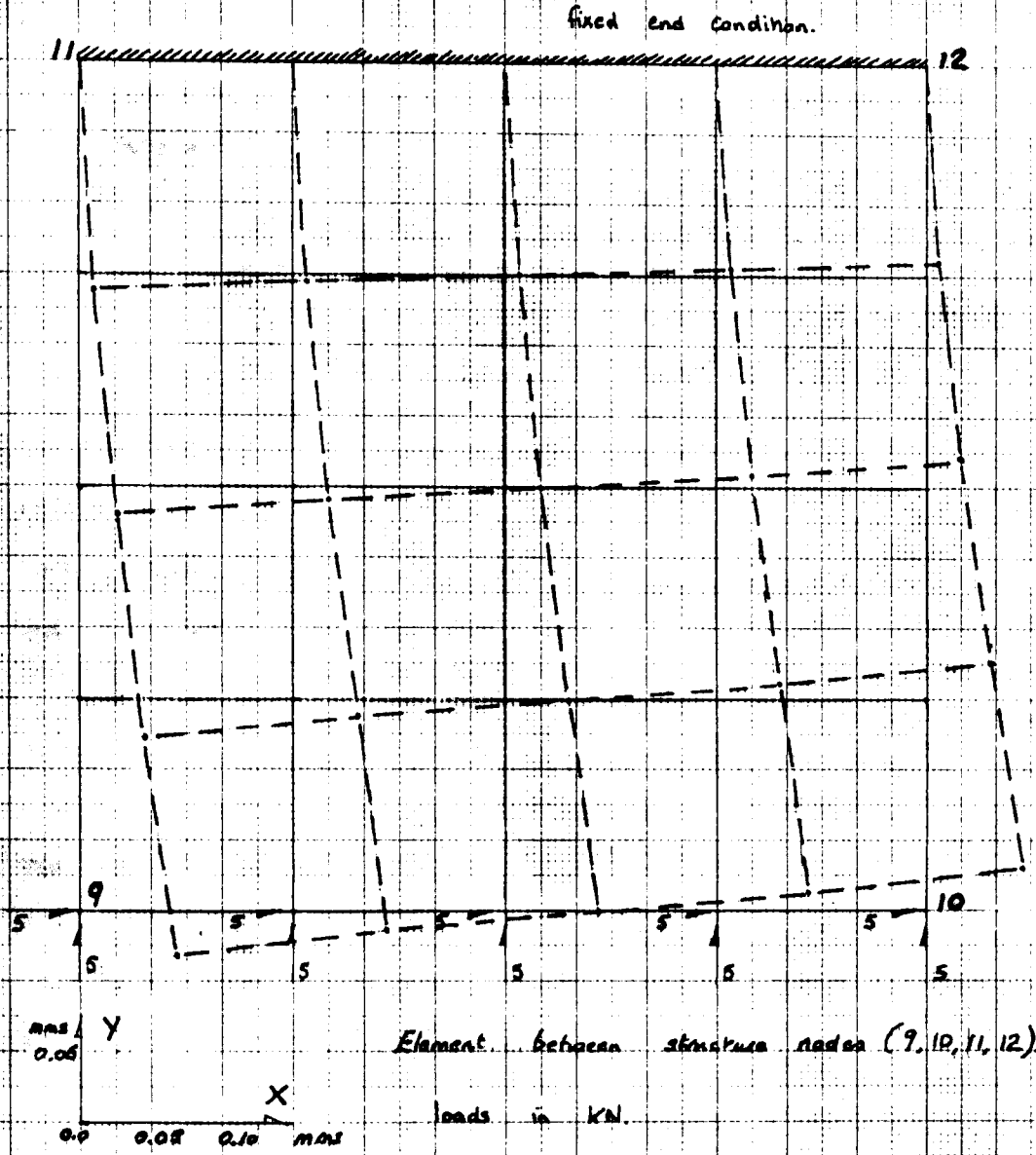


Fig. 5.20 Displacement of slab element loaded in it's own plane.

TABLE 5.9.

Joint	DEFX	DEFY
5	681	-292
6	681	292
9	681	-292
10	681	292

deflections - mms x 10^4

2) Wall deformation.

Fig. 5.20 shows diagramatically the deformation of the wall under the loads described.

Comments on Example 4.

This example is given to show how the program can be used to analyse diaphragm elements for either in-plane or out of-plane forces. Note how the structure axes have been rotated to enable the analysis to be performed.

5.14 Verification and Limitation of DECK1 program.

The DECK1 program has undergone a limited testing procedure to justify the validity of results. However, it is still very much a prototype program and cannot be considered as fully tested. Verification was undertaken in two stages by comparison

(a) with known results

and (b) with other tested programs.

The Tee-beam and Rigid Orthogonal Space Frame programs described in earlier sections (2&3) were used to establish that the frame results were valid. Floor solutions were tested by

simulating known cases so that comparisons could be made. A further check on the total structure analysis was made by comparing results on a test frame analysed by Majid and Williamson[4]. The frame was a table-like structure and the solutions obtained showed good agreement allowing for small variations in material properties. Also, the conclusion that the increase in stiffness of a floor-clad structure over that of the bare frame was of the order of 28% was reinforced.

Certain limitations have been placed on the node numbering system adopted throughout the program. For structure nodes the program requires that numbering must follow a sequential system from floor to floor i.e. one floor must be totally numbered before moving on to the next. A similar assumed system is in operation for the floor mesh numbering. Here if mesh nodes are numbered in the X-direction the input will expect elements in rows in the same direction.

Hardware restrictions will, of course, limit the size of structure which can be analysed and the size of floor mesh that can be handled. The input and output systems are somewhat rigid, not allowing much choice to the user. However, improvements in this direction can be easily implemented and in the case of the output a system for some kind of user choice has been added.

To illustrate the capability of the program a 3 storey structure was solved. It consisted of a grid of 12 columns supporting floors split into 36 elements, the total number of unknown displacements being 972. The machine used was an Elliot 4130 where the store required for systems, program and operation was 49½k. The total time for run and output

was 12 minutes which consisted of printed input data and total output, comprising displacements and forces for 36 column members, 27 beam elements per floor and 36 floor elements. It is not possible to state the capacity of the program as this greatly depends on the type of structure to be analysed, however, the foregoing example does indicate the program's scope and shows its affinity for the multi-storey nature of structures.

5.15 References.

1. R. K. Livesley - 'Matrix Methods of Structural Analysis' Pergamon Press London 1964.
2. M. H. E. Larcombe - 'The factorisation of member stiffness matrix to reduce storage requirements, and some comments on restraints' Paper presented at the International Symposium on Computer-Aided Structural Design at Warwick University, 10-14 July 1972.
3. MOT - 'Suite of Bridge Design Analysis Programs', Program BECP/1. MOT/CIRIA - 'Finite Element Package for the Analysis of Reinforced Concrete Slab Bridge Decks', Vol. User Manual, Vol. Report, 1st Edition, May 1969.
4. K. I. Majid & M. Williamson - 'Linear Analysis of Complete Structures by Computers'. Proceedings of I.C.E. Vol. 38, Oct. 1967, 247-266.

6. COMPARATIVE ELASTIC ANALYSIS OF TEST FRAMES.

6.1 Introduction.

In this section the results from test frames analysed by the programs described earlier in the text are presented. The three programs are TEE1, the Tee-beam analysis program, RIGIDORT, the rigid orthogonal space frame program and DECK1, the program described in section 5. The three programs illustrate the different approaches available to the analyst when using computers to obtain elastic analysis solutions.

The DECK1 program is further used to investigate the effect of floor mesh grading on two simple test frames and also to carry out investigations into the possibility of using a sub-framing technique in three-dimensions similar to that used for plane frames[1,2]. For this purpose results for sub-frame analyses are compared with those for the total test frame. This investigation was restricted in its size but does provide interesting information in the technique of sub-framing.

The results are presented in tabular or graphical form as appropriate, and where necessary comments and worthwhile relationships have been given. A general discussion on this section will be given later in section 7.

6.2 Test 1.

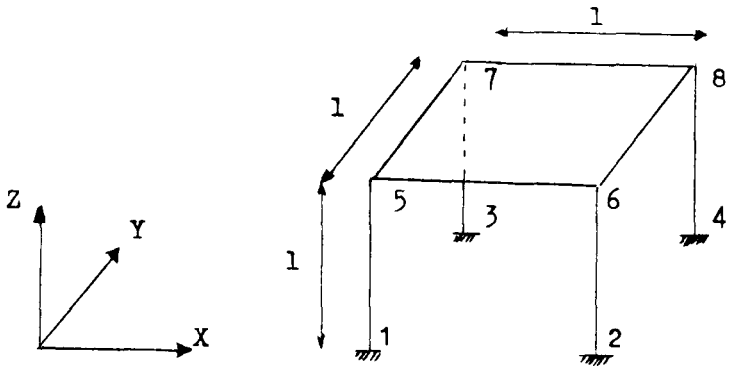


Fig. 6.1. Diagram of the Test Frame Structure.

Properties: l = 6000mms floor thickness = 120mms

Young's Modulus E = 13.78 KN/mm^2

Shear Modulus G = 5.512 KN/mm^2

Poisson's Ratio ν = 0.15

Column size 200 x 200 mms

Members 5 - 6, 5 - 7, 6 - 8, and 7 - 8 are used as beams in tests on beam supported floors and for beams in the RIGIDORT tests.

TABLE 6.1 - Testing Schedule.

Method of Solution for Elastic Analysis	Structure Data.	Loading Cases.
1. Plane Frame Analysis With Rectangular or Tee-beams.	A 'half-structure' frame. 	a) Uniformly distributed load* on member 5 - 6. b) Point load at joint 5 in the X direction.
2. Skeletal Space Frame Analysis With Rectangular Beams.	As illustrated in Fig. 6.1 but without a floor deck.	a) Equivalent UDL vector* b) Point loads at joints 5 and 7 in the X direction. c) Point load at joint 5 in the X diirection.
3. Decked Frame Solution - Full Rigid Frame With Floor and Rectangular Beams.	As illustrated in Fig. 6.1.	a) UDL on floor area. b) Point loads at joints 5 and 7 in the X direction. c) Point load at joint 5 in X direction.

* An equivalent loading to that of the uniform load on the floor area is applied to the plane frame and to the skeletal frame. For the plane frame, a line load equal to half the total floor load is used. For the skeletal frame, the loading equivalent is as described in section 3, (3.4).

6.3 Results for Test 1.

6.3.1 Results for loading case (a) - uniformly distributed load of 1KN/M^2 on the floor area.

TABLE 6.2 - Displacement of node 5 of test frame.

Program	Beam Data (mms) (depth x breadth)	ROTX	ROTY	ROTZ	DEFX	DEFY	DEFZ
a) TEE1	120 x 200	-	66	-	184	-	-980
RIGIDORT	120 x 200	-66	66	0	184	184	-980
"	120 x 300	-63	63	0	117	117	-980
"	120 x 500	-58	58	0	64	64	-980
"	120 x 700	-53	53	0	42	42	-980
DECK1	-	-24	24	0	9	9	-980
b) TEE1	320 x 200	-	24	-	25	-	-980
"	320 x 200 Tee-beam with floor 120 deep, flange 680 wide.	-	16	-	10	-	-980
RIGIDORT	320 x 200	-24	24	0	25	25	-980
DECK1	200 x 200	-20	20	0	6	6	-980
RIGIDORT	200 x 200	-49	49	0	82	82	-980
TEE1	200 x 200	-	49	-	82	-	-980

Rotations - radians x 10^4.

Deflections - mms x 10^4.

TABLE 6.3 - Column End Forces for Member 1 - 5 of Test Frame.

Program	Beam Data (mms) (depth x breadth)	MY1=MZ1	MY2=MZ2	AX1	SY1	SZ1
a) RIGIDORT	120 x 200	4057	8120	9.0	2.03	-2.03
"	120 x 300	3869	7742	9.0	1.94	-1.94
"	120 x 500	3540	7082	9.0	1.77	-1.77
"	120 x 700	3263	6526	9.0	1.63	-1.63
DECK1	-	1396	2792	9.0	0.70	-0.70
b) TEE1	320 x 200	1480	2950	9.0	-	-0.74
"	320 x 200 Tee-beam with floor 120 deep, flange 680 wide.	1010	2010	9.0	-	-0.50
DECK1	200 x 200	1183	2367	9.0	0.60	-0.60
RIGIDORT	200 x 200	2998	5999	9.0	1.50	-1.50
TEE1	200 x 200	3000	6000	9.0	-	-1.54

forces - KN or KNmm.

6.3.2 Results for loading case (b) - horizontal sway load on joints 5 and 7 of 5KN in the X direction.

TABLE 6.4 - Displacement for nodes 5 and 6 of test frame.

Program	Beam Data (mms) (depth x breadth)	NODE 5			NODE 6		
		ROTY	DEFX	DEFZ	ROTY	DEFX	DEFZ
a) TEE1	120 x 200	0.0107	56.53	0.0154	0.0107	56.48	-0.0154
RIGIDORT	120 x 200	0.0107	56.62	0.0154	0.0107	56.58	-0.0154
"	120 x 300	0.0083	49.56	0.0180	0.0083	49.53	-0.0180
"	120 x 500	0.0058	41.92	0.0208	0.0058	41.90	-0.0208
"	120 x 700	0.0044	37.86	0.0233	0.0044	37.84	-0.0233
DECK1	-	0.0035	35.15	0.0233	0.0035	35.14	-0.0233
b) TEE1	320 x 200	0.0010	27.40	0.0262	0.0010	27.38	-0.0261
RIGIDORT	320 x 200	0.0010	27.46	0.0262	0.0010	27.44	-0.0261
DECK1	200 x 200	0.0016	29.28	0.0255	0.0016	29.28	-0.0255
TEE1	200 x 200	0.0035	35.02	0.0234	0.0035	35.00	-0.0233
RIGIDORT	200 x 200	0.0035	35.11	0.0233	0.0035	35.08	-0.0233

Deflections - mms

Rotations - radians

TABLE 6.5 - Column forces for member 1 - 5.

Program	Beam Data (mms) (depth x breadth)	MY1	MY2	AX1	SZ1
a) RIGIDORT	120 x 200	-10770	-4240	-1.41	2.5
"	120 x 300	-10050	-4960	-1.65	2.5
"	120 x 500	-9270	-5730	-1.91	2.5
"	120 x 700	-8850	-6150	-2.05	2.5
DECK1	-	-8580	-6420	-2.10	2.5
b) TEE1	320 x 200 Tee-beam with floor 120 deep, flange 680 wide.	-7760	7370	-2.43	2.52
DECK1	200 x 200	-7980	7020	-2.33	2.50
RIGIDORT	320 x 200	-7800	7210	-2.40	2.50

forces - KN or KNmm.

6.3.3 Results for loading case (c) - A twisting load of 5KN applied at node 5 of the test frame.

TABLE 6.6 - Displacement of Nodes 5 and 7 of Test Frame.

Program	Beam Data (mms)(depth x breadth)	NODE 5				NODE 7			
		ROTY	ROTZ	DEFX	DEFZ	ROTY	ROTZ	DEFX	DEFZ
a) RIGIDORT	120 x 200	85	-28	45.88	0.0113	23	-28	10.71	0.0043
"	120 x 300	64	-26	39.45	0.0124	19	-26	10.11	0.0055
"	120 x 500	43	-24	32.74	0.0136	14	-24	9.15	0.0072
"	120 x 700	33	-23	29.24	0.0140	12	-23	8.61	0.0083
DECK1	-	27	-0	26.52	0.0117	8	-0	8.63	0.0116
b) RIGIDORT	200 x 200	25	-23	26.87	0.0143	10	-23	8.22	0.0093
"	320 x 200	8	-20	20.29	0.0143	3	-20	7.20	0.0120
DECK1	200 x 200	8	0	15.13	0.0127	8	-0	14.15	0.0127

Deflections - mms

Rotations - radians x 10^4

6.4 Comments on Test 1.

In TABLE 6.2 (a), where the frame is subjected to a uniform floor load, the effect of increasing the edge beam width in order to compensate for the higher bending and in-plane stiffness of the floor slab is illustrated. The results show that the required width would be greater than 25% of the total floor width in order to produce a similar result using beams instead of the floor slab. It is reasonable to assume that in this test case the slab would span in two directions. The loading vector used for the RIGIDORT test takes account of this two-way spanning action. However, in the TEE1 test a uniform line load was used where a triangular load would have been more realistic for the calculation of beam moments. Yet, as is shown in TABLE 6.2 (a), the displacements are in good agreement and if triangular loading was to be used the column stiffnesses would have to be adjusted to account for the reduction in axial column load. Similar results for increasing edge beam widths are shown by the column forces given in TABLE 6.3 (a). As is shown in TABLE 6.2 (b), when supporting beams were added to the DECK1 test frame, the difference in rigidity was not so great as for the non-beam case due to a large proportion of bending being taken by the edge beams, The use of Tee-beam sections leads to an improvement in results, as shown in TABLE 6.7, where some calculated end moments for the plane frame test case have been given for both uniform and triangular loading, the beam being considered as a flanged beam. The values were obtained using

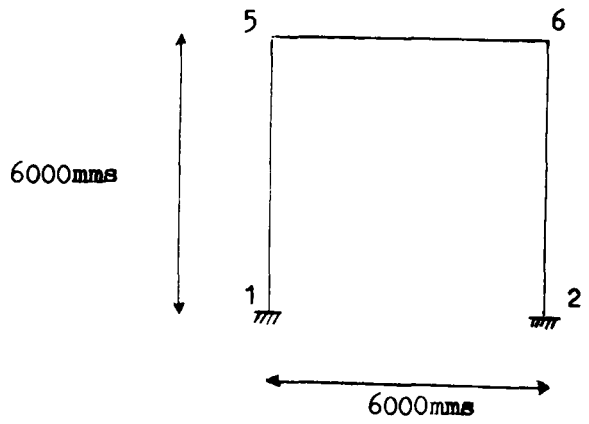


Fig. 6.2 Test Frame for Moment Distribution.

Member 5 - 6, a 200 x 200mm beam, is considered as a flanged beam with flange as specified below and floor depth of 120mms. The columns are 200 x 200mms in cross-section.

TABLE 6.7 End Moments for Columns 1 - 5 of Test Frame in Fig. 6.2.

Loading Mode	Flange Size (mms)	Moment at 1 (KNmms)	Moment at 5 (KNmms)
Uniform load of 3KN/M	500	2530	4950
"	1000	2110	4190
Triangular Loading, total load = 9KN	500	1580	3160
"	1000	1320	2620
Uniform load of 1KN/M^2	from DECK1 results (cf TABLE 6.3(b))	1183	2367

a moment distribution and it is apparent that the best agreement of column moments, compared to those given in section 6.3, is for the largest flanged beam subjected to a triangular loading, the flange size being 1/6th of the actual floor span.

The second and third loading cases illustrate the high in-plane stiffness which the floor slab provides for the frame, but the effect is not so marked as in the flexural case discussed previously, especially for the beam supported floor. The limited deformation of the floor plan area, shown by the displacements given in TABLE 6.6, demonstrates this extra in-plane stiffness compared with the bare frame. As before, an increase in beam width leads to an improvement in these results.

6.5 Test 2 - Effect of Mesh Concentration.

Two frames carrying a uniform deck load have been analysed using the DECK1 program for the purpose of investigating the influence of floor mesh grading on the precision of results.

6.5.1 Frame 1

The first test frame, the same as that given in Fig. 6.1 for Test 1, was subjected to a uniformly distributed floor load of $1KN/M^2$. The element mesh concentrations used in the test were 9, 16 and 25. The results of the tests, illustrating the variation of structure displacements and forces with floor mesh concentration, are presented in the following tables and graphs. The frame has been tested with and without supporting

edge beams of 200 x 200mms cross-section.

TABLE 6.8 Displacements of Node 5 of Test Frame.

Number of Mesh Nodes	ROTX	ROTY	ROTZ	DEFX	DEFY	DEFZ
(a) Without supporting beams						
16	-23	23	0	8	8	-980
25	-24	24	0	9	9	-980
36	-24	24	0	10	10	-980
(b) With supporting beams						
16	-19	19	0	5	5	-980
25	-20	20	0	6	6	-980
36	-21	21	0	7	7	-980

deflections in mms, rotations in radians, all results x 10^4.

TABLE 6.9 Column Forces for Member 1 - 5 of Test Frame.

Number of Mesh Nodes	MY1=MZ1	MY2=MZ2	SZ1=SZ2	AX1
(a) Without supporting beams				
16	1396	2792	0.7	9.0
25	1443	2886	0.7	9.0
36	1450	2901	0.7	9.0
(b) With supporting beams				
16	1183	2367	0.6	9.0
25	1248	2495	0.6	9.0
36	1277	2554	0.6	9.0

Forces in KN or KNmm

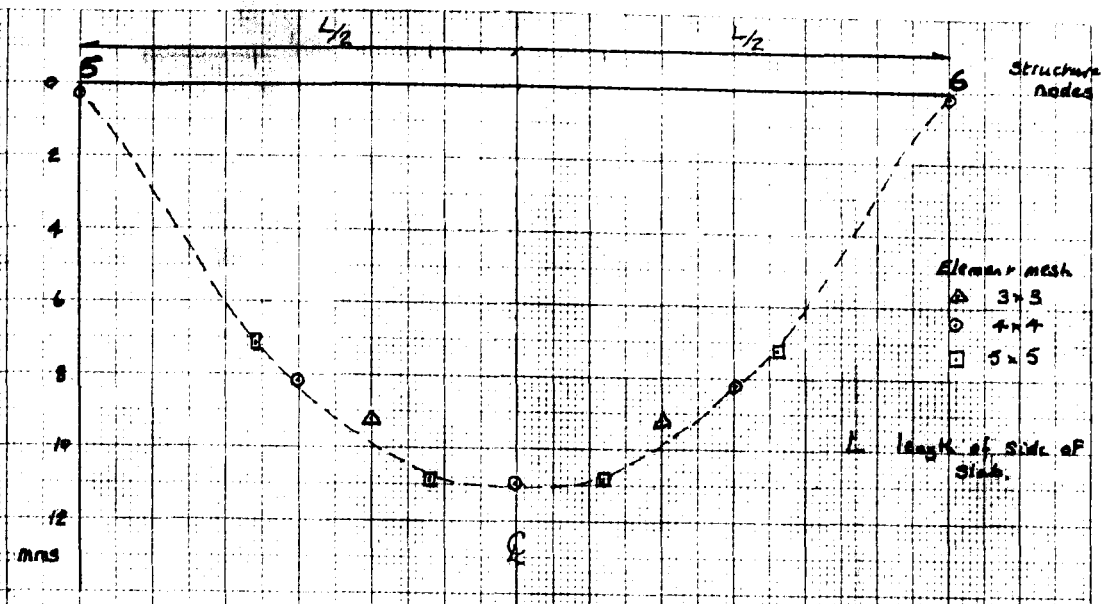


Fig. 6.3 (a) Variation of vertical deflection of floor without supporting beams with respect to mesh size for Test Frame 3

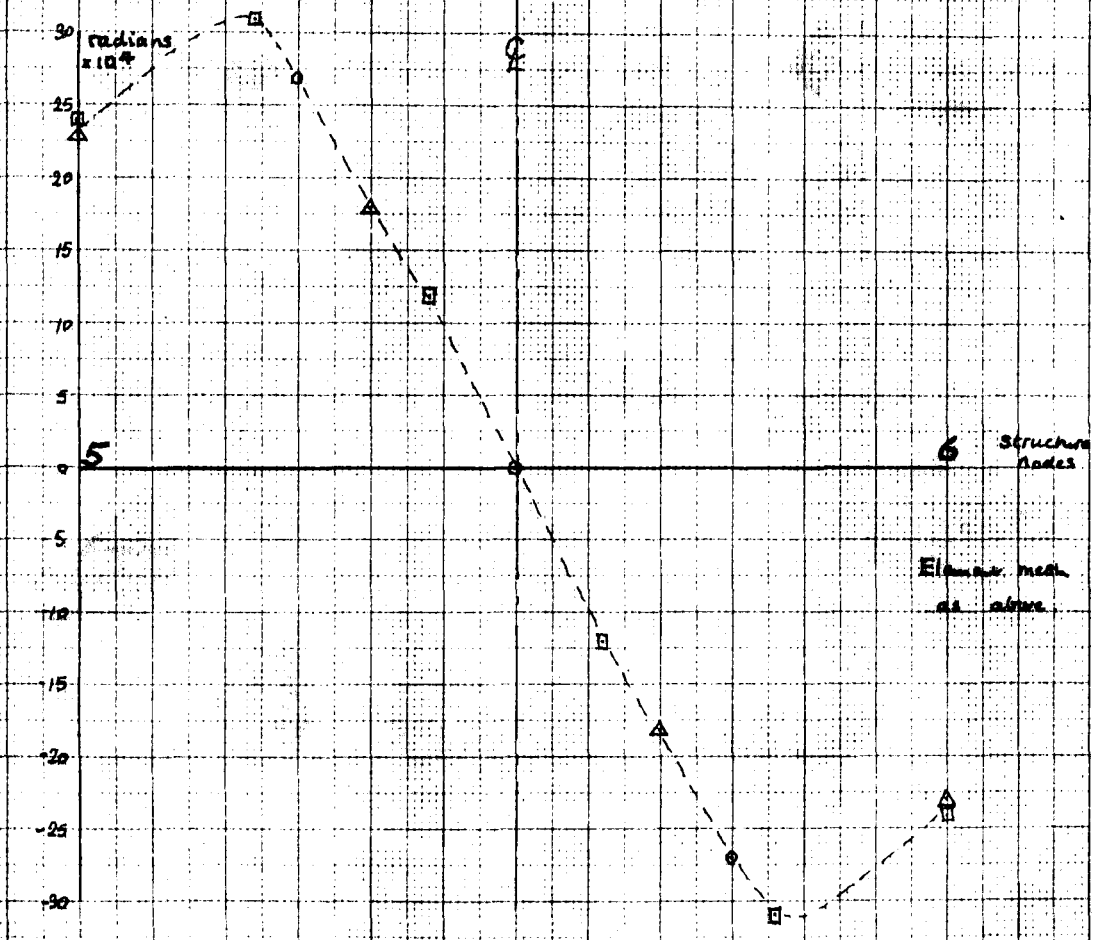


Fig. 6.3 (b) Variation of rotation θ_y along X-direction edge of floor of Test Frame 3 with respect to mesh size.

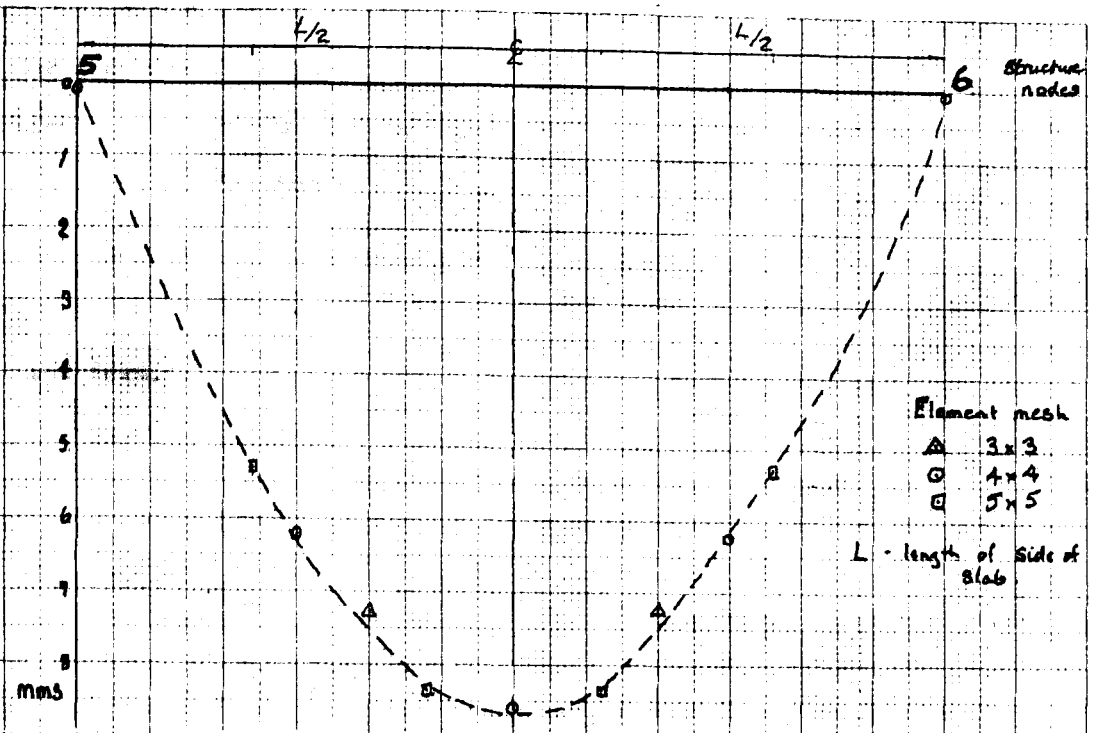


Fig. 6.4 (a) Variation of vertical deflection of floor slab along x edge with mesh size for Test Frame 3 with supporting beams.

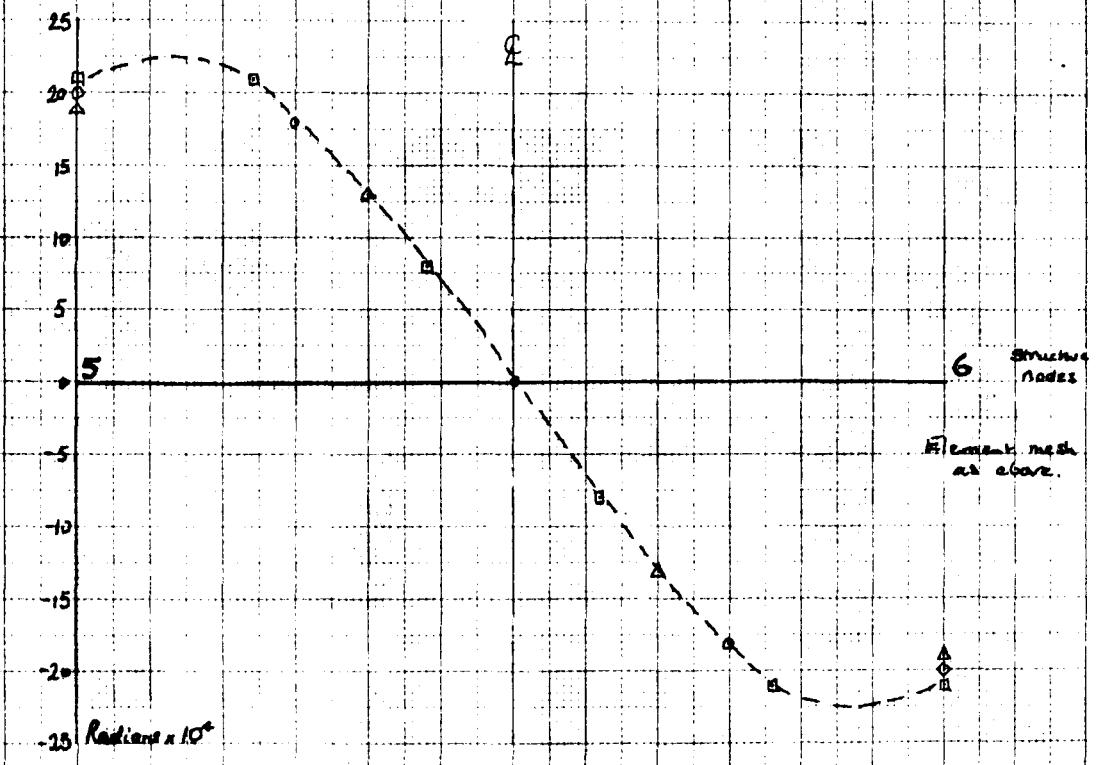


Fig 6.4 (b) Variation of rotation θ_y along x edge of floor slab with mesh size for Test Frame 3 with supporting beams.

6.5.2 Frame 2.

The test frame is illustrated in Fig. 6.5 The uniform floor loading was 10KN/M^2 and two mesh gradings were used:-

(a) 3x7 mesh nodes

and (b) 5x7 mesh nodes.

The results for the test are shown in table, graph and contour plot form.

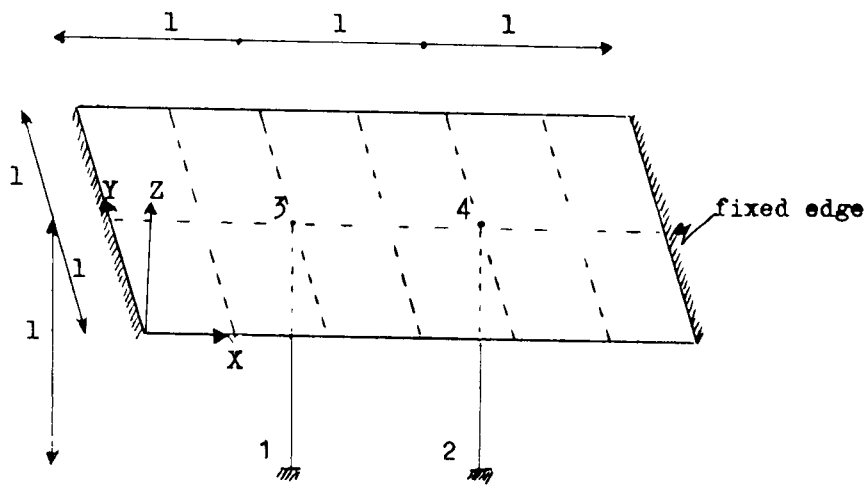


Fig. 6.5 Diagram of Test Frame Structure.

Properties: 1 = 1000mms floor thickness = 120mms

Young's Modulus E = 13.78 KN/mm^2

Shear Modulus G = 5.512 KN/mm^2

Poisson's Ratio ν = 0.15

column size 200 x 200mms

TABLE 6.10 Displacements of Structure Node 3.

Number of Mesh Nodes	ROTX	ROTY	ROTZ	DEFX	DEFY	DEFZ
3 x 7	0	1	0	12	0	-273
5 x 7	0	0	0	2	0	-287

Deflections in mms

Rotations in radians, all results x 10^4

TABLE 6.11 Column Member-end Forces for Member 1 - 3.

MESH	MY1	MY2	AX1	SZ1
21	183.9	380.4	15.00	-0.06
35	169.7	341.4	15.83	-0.51

forces in KN or KNmm

6.6 Comments on Test 2 - Effect of Mesh Concentration.

The results obtained for the table-like structure showed that a convergence is achieved by the use of increasing mesh density. However, the actual improvement in precision for the column forces given in TABLE 6.9 is not substantial. Here the greatest divergence in column end moment for the three mesh sizes is less than 10%. Similar results are provided by the analysis of the second frame where the divergence of displacement and force

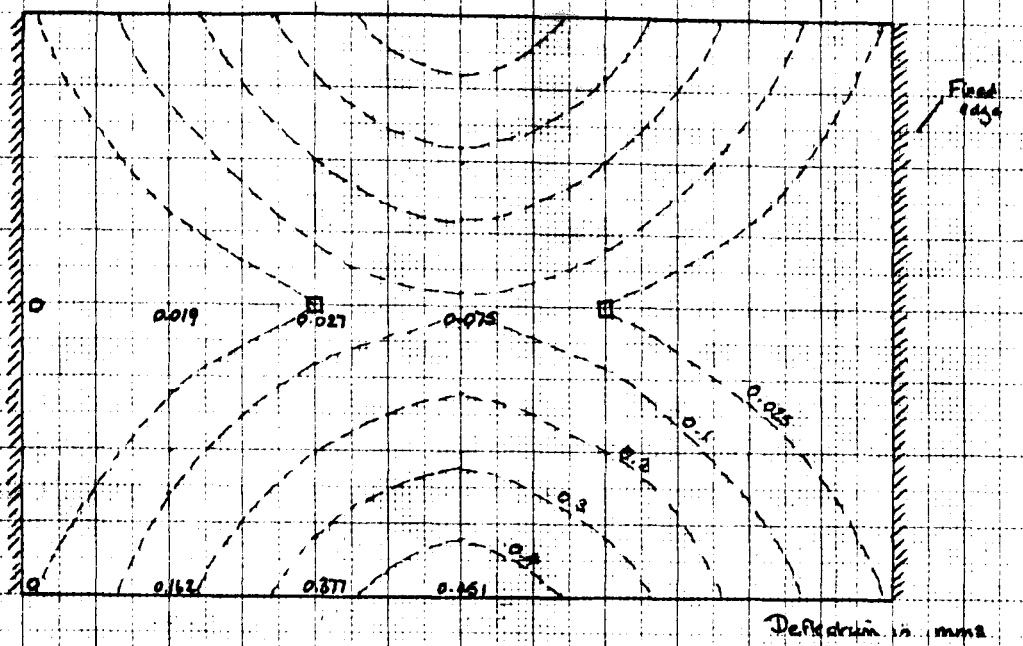


Fig. 6.6 Contour plot of vertical deflection of bridge deck of Test Frame 4 for a floor mesh of 21 nodes.

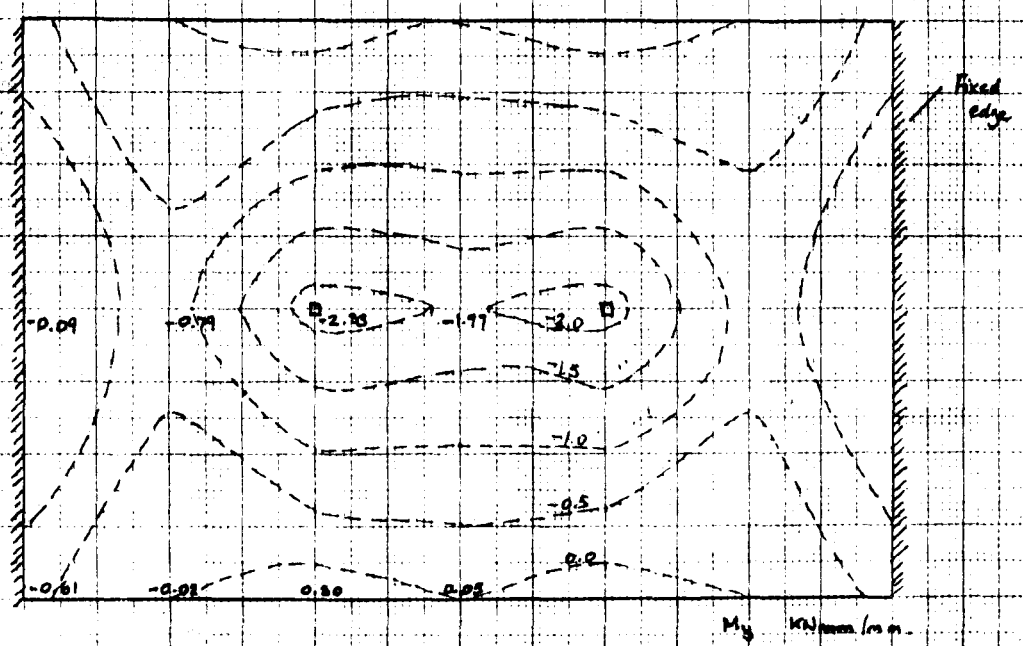


Fig. 6.7 Contour plot of plate moment My of bridge deck of Test Frame 4 for a floor mesh of 21 nodes.

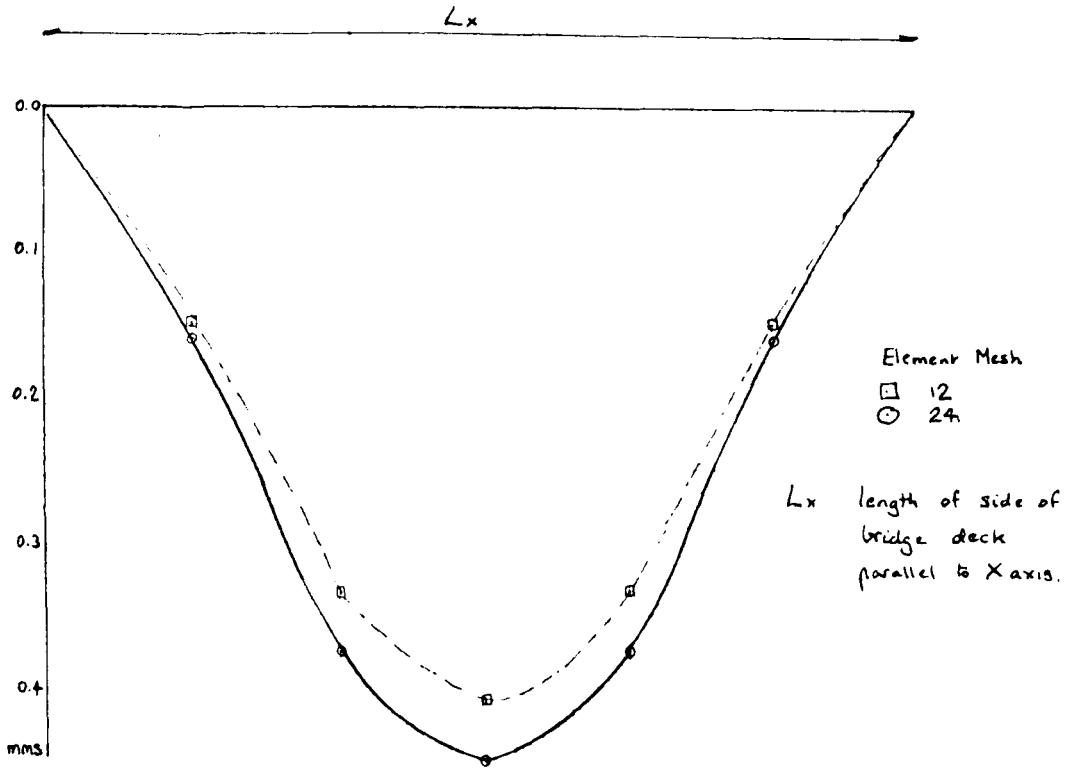


Fig 6.8 (a) Variation of vertical deflection, along X - edge of deck of Test Frame 4, with mesh size.

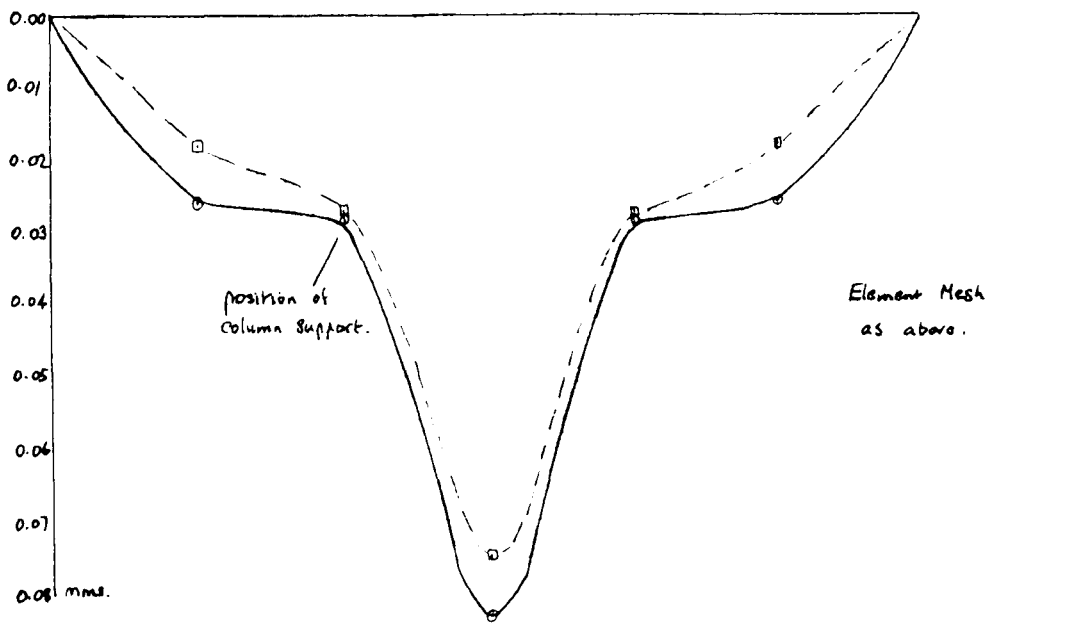


Fig. 6.8 (b) Variation of vertical deflection along X - Centre line of deck of Test Frame 4 with mesh size.

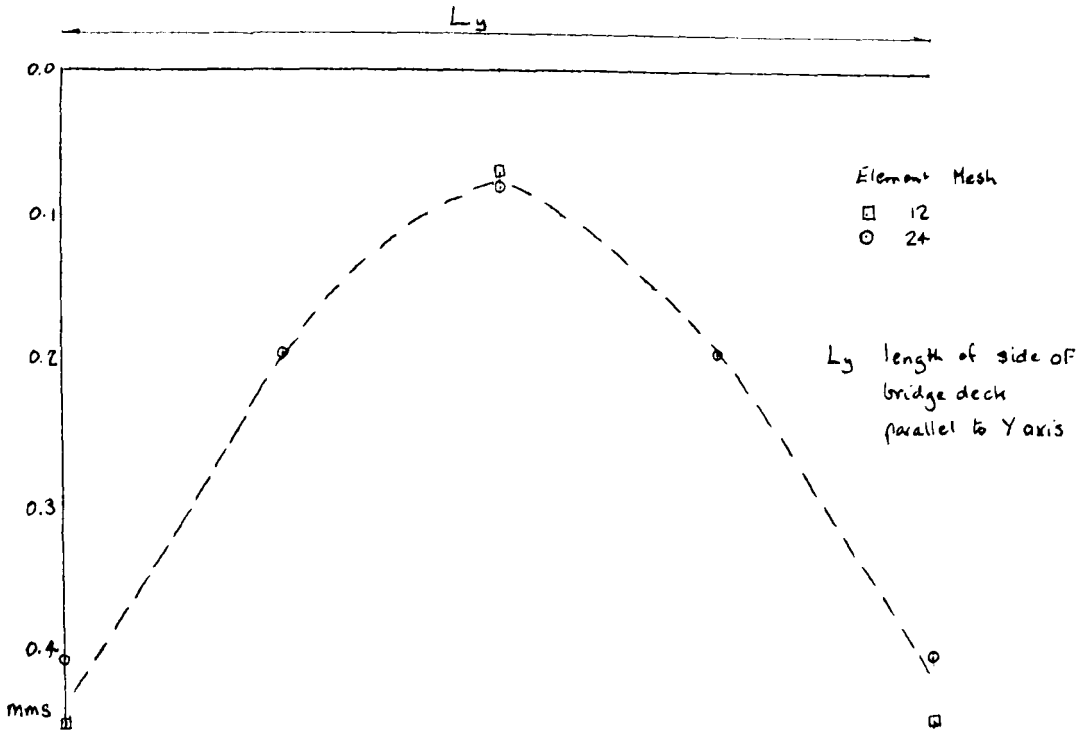


Fig. 6.8 (c) Variation of vertical deflection along y-centre line of bridge deck of Test Frame 4 with mesh size.

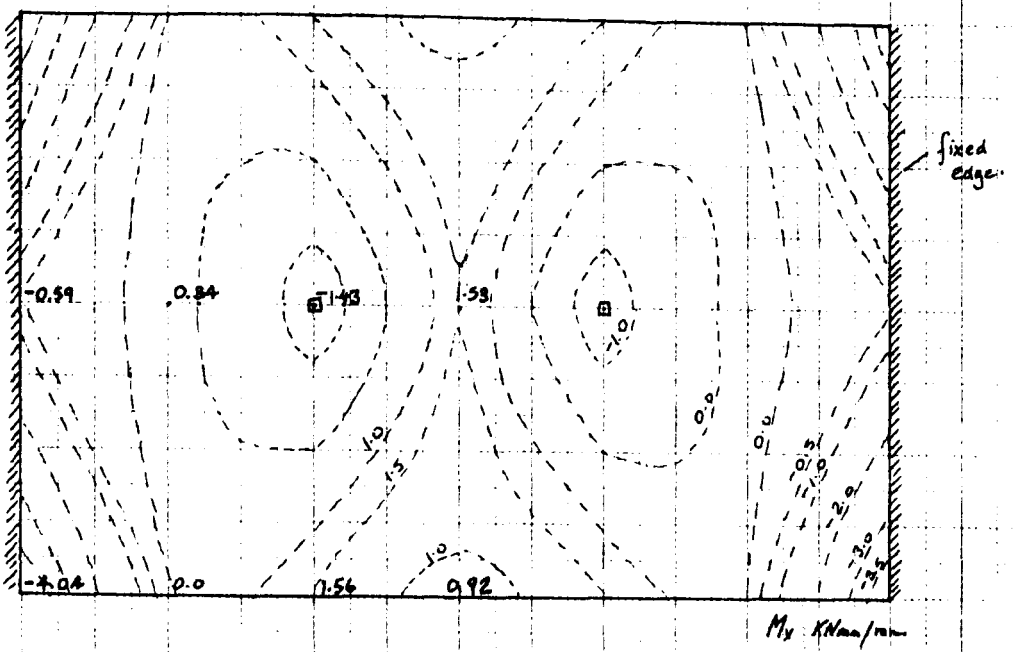


Fig 6.9 (a) Contour plot of floor moment Mx for bridge deck of Test Frame 4 with mesh nodes = 21

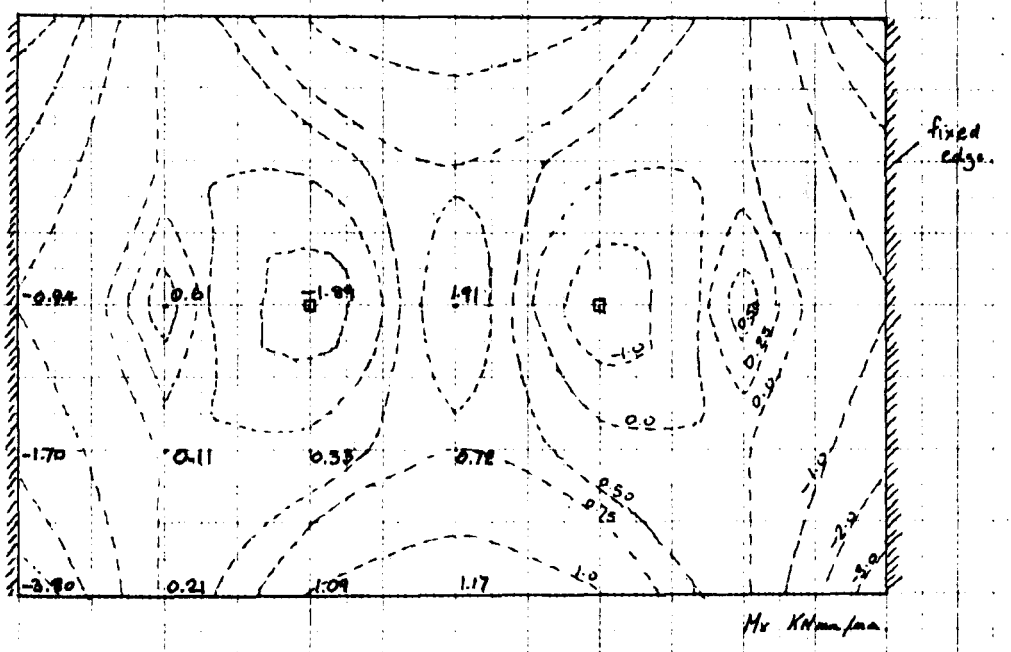


Fig 6.9 (b) Contour plot of floor moment Mx for bridge deck of Test Frame 4 with mesh nodes = 35

values was greater.

The effect of varying the element mesh on the floor solution is illustrated by the graphs shown in Figs. 6.3 and 6.4. Again it can be seen that there is only a small divergence of results and that the increase in accuracy in using a finer mesh is not large. In the second test, however, the divergence of results is significant and the use of a finer mesh does seem desirable if floor behaviour is being investigated. This is borne out by the graphs for displacement, as shown in Figs. 6.8 (a), (b) and (c), for the floor deck.

The contour plots shown in Figs. 6.6, 6.7 and 6.9 demonstrate how floor information can be interpreted and how the size of mesh can limit the accuracy of the plotting. This is especially true of the floor moments as shown in Fig. 6.9.

6.7 Test 3 - Sub-framing.

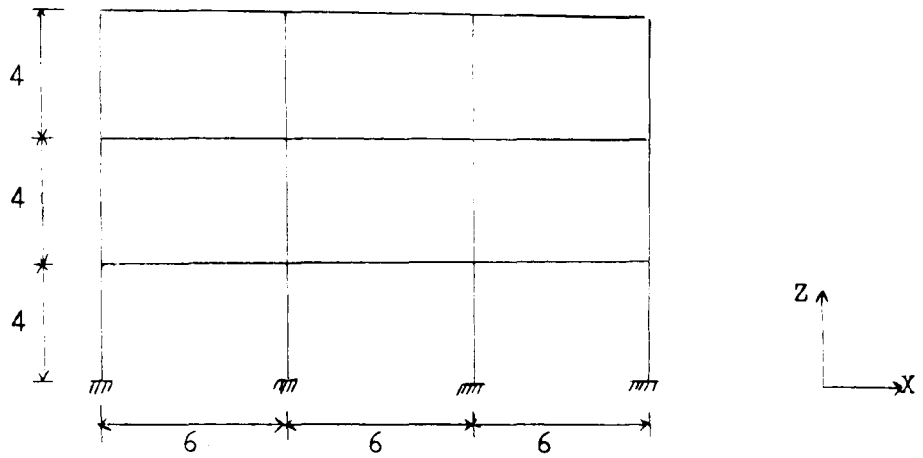
A preliminary investigation into the possibility of using three-dimensional sub-frames for the determination of member end forces for beams of a space frame was undertaken. The sub-frame was to account for three-dimensional behaviour of the structure. The test frame chosen at this stage in the investigation was a regular, symmetric structure in order that basic problems were clear, which might not be the case if the frame was irregular. The starting point was to take the beam in question and form the sub-frame by considering the adjacent members, fully fixed at their extremities, plus the accompanying floor slab. However,

the fixity of the slab at the boundary was undecided, as is shown by the following results. The test consisted of setting up sub-frames, which were analysed using the DECK1 program, to account for internal and boundary beam members. The sub-frame results were compared with those for the full frame, which was also analysed using DECK1. The results for both sub-frame and full frame are presented below, where the fixity of the floor boundary for the sub-frames has been varied. In this test run only local loading around the member can be considered and for this reason only uniform loading has been applied to the frames.

6.8 Test 3 - Results on Sub-framing.

6.8.1

The full space structure consisted of three frames as shown in Fig 6.10 interconnected by the floor slab at a bay width of 5 metres. The frame was analysed for a floor loading of $15KN/M^2$ and the results are shown, where required for comparison with the sub-frame solutions. Each floor area was divided into 36 finite elements.



Dimensions in metres.

Fig. 6.10 Section Through Space Frame Used for Sub-frame Testing.

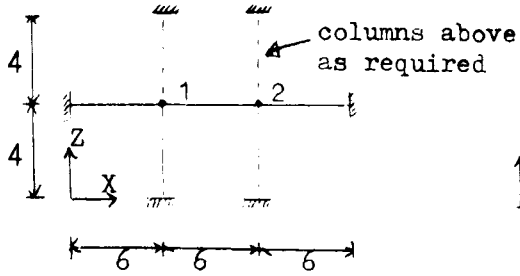
Properties:	Area (M^2)	I_y (M^4)	I_z (M^4)	J (M^4)
Beams:	.18	.0054	.0054	.0054
Columns:	.09	.002025	.002025	.002025

Bay width = 5 metres

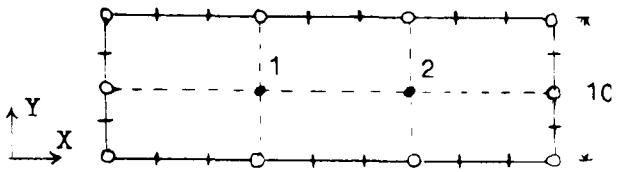
Floor thickness = 200mms

6.8.2 Internal member sub-frames.

The sub-frame for roof and lower storey members are both shown in Fig. 6.11. The member end forces for both cases are given in TABLE 6.12.



Floor Plan.



o - column boundary node
+ - floor boundary node

Dimensions in metres,
properties for beams and
columns as for full frame.

Key to floor boundary fixity.

Mode A :- o - fully restrained; + - fully restrained.

B :- o - " " + - Z deflection restrained.

C :- o - Z deflection restrained; + - Z " "

Fig 6.11 Sub-frame for Internal Member of Space Frame.

TABLE 6.12 - End Forces for Member 1 - 2 of Fig 6.11.

Floor Boundary Fixity Mode	END 1			END 2		
	SZ1 (KN)	MX1 (KNM)	MY1 (KNM)	SZ2 (KN)	MX2 (KNM)	MY2 (KNM)
(a) Lower storey members						
A	101.8	0	-131.6	101.8	0	131.6
B	116.3	0	-152.4	116.3	0	152.4
C	128.1	0	-178.9	128.1	0	178.9
Full frame analysis	108.6	0	-141.0	108.6	0	141.0
(b) Roof members						
B	114.8	0	-147.9	114.8	0	147.9
Full frame analysis	112.9	0	-149.3	112.9	0	149.3

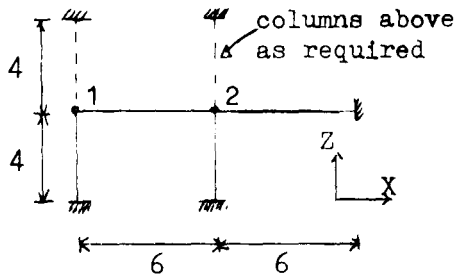
For key to floor boundary fixity modes see Fig. 6.11

6.8.3 Boundary member sub-frames.

(a) One end of the beam at the boundary.

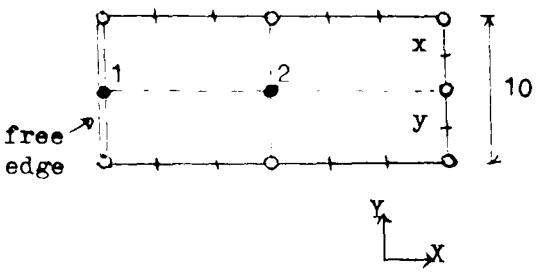
The sub-frames for both (a) roof and (b) lower storey cases are shown in Fig. 6.12(a) and the comparative results for this type of boundary condition are shown in TABLE 6.13.

Fig 6.12(a) Sub-frame for Boundary Member of Space Frame where One End of the Beam is at the Boundary.



Dimensions in metres, properties for members as for full frame.

Floor Plan.



o - column boundary node.
+ - floor boundary node.

Key to edge fixity.

Mode A :- o - fully restrained; + - fully restrained.

B :- o - " " + - Z deflection restrained.

C :- o - fully restrained, except at free edge where only restrained from vertical displacement.

D :- o - fully restrained; + - nodes x,y restrained from vertical displacement.

E :- o - Z deflection restrained; + - Z deflection restrained.

TABLE 6.13 End Forces for Member 1 - 2 of Sub-frame of Fig. 6.12(a).

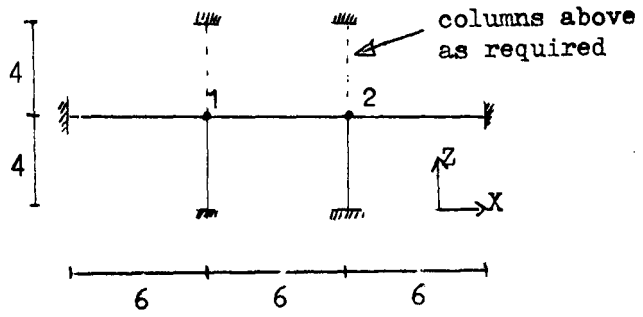
Boundary Edge Fixity Mode	END 1			END 2		
	SZ1 (KN)	MX1 (KNM)	MY1 (KNM)	SZ2 (KN)	MX2 (KNM)	MY2 (KNM)
(a) Roof members						
A	73.8	0	-42.7	110.9	0	142.4
E	82.4	0	-31.4	152.4	0	217.8
Full frame analysis	95.9	0	-74.3	122.0	0	148.4
(b) Lower storey members						
A	81.2	0	-64.6	118.8	0	147.3
B	95.5	0	-75.3	130.6	0	173.9
C	97.6	0	-76.1	132.0	0	175.9
D	105.7	0	-84.3	135.3	0	181.0
Full frame analysis	107.3	0	-117.4	112.3	0	137.4

For key to boundary edge fixity modes see Fig 6.12(a).

(b) One edge of the beam along the boundary.

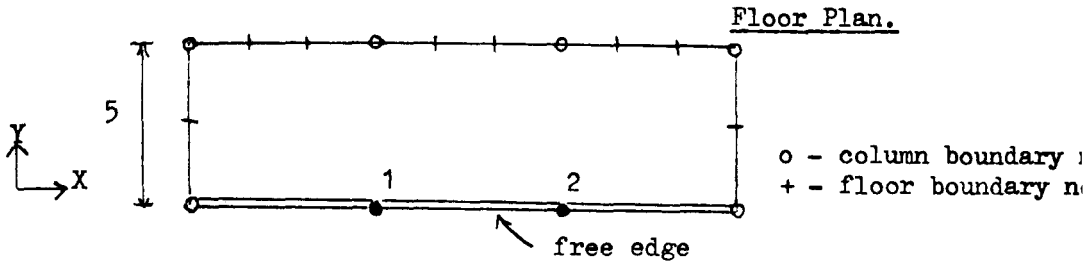
The sub-frame for both (a) roof and (b) lower storey members is illustrated in Fig 6.12(b) and the test results in TABLE 6.14.

Fig 6.12(b) Sub-frame for Boundary Members of Space Frame where One Edge of the Beam is Along the Boundary.



Dimensions in metres, properties of beams and columns as for full frame

Fig 6.12(b) continued.



Key to edge fixity.

Mode A :- o - fully restrained; + - fully restrained.

B :- o - " " + - Z deflection restrained.

C :- o - Z deflection restrained; + - Z " "

TABLE 6.14 - End Forces for Member 1 - 2 of Sub-frame of Fig. 6.12(b).

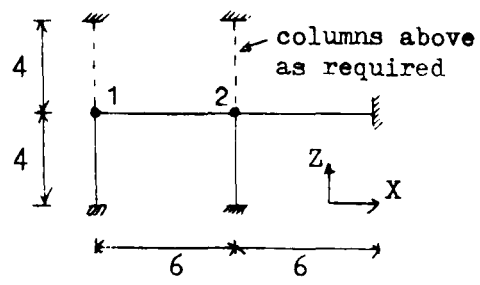
Boundary Edge Fixity Mode	END 1			END 2		
	SZ1 (KN)	MX1 (KNM)	MY1 (KNM)	SZ2 (KN)	MX2 (KNM)	MY2 (KNM)
(a) Roof members						
A	52.5	7.9	-67.3	52.5	7.9	67.3
B	59.2	13.5	-77.1	59.2	13.5	77.1
C	64.9	8.7	-93.1	64.9	8.7	93.1
Full frame analysis	56.5	16.5	-76.5	56.5	16.5	76.5
(b) Lower storey members						
A	53.3	11.4	-69.5	53.3	-11.4	69.5
Full frame analysis	57.4	23.2	-75.2	57.4	-23.2	75.2

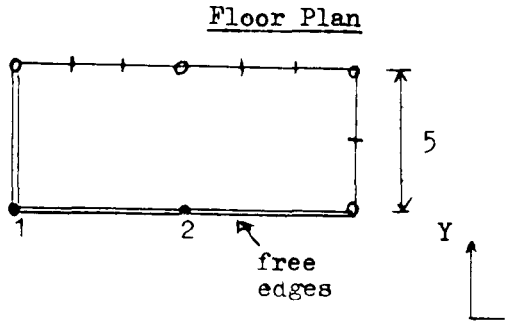
For key to boundary edge fixity see Fig 6.12(b).

(c) Beam at the corner of the boundary.

The sub-frame for (a) roof and (b) lower storey members is given in Fig 6.12(c) and the corresponding member end forces in TABLE 6.15.

Fig 6.12(c) Sub-frame for Boundary Members of Space Frame with the Beam at the Corner of Boundary.





Dimensions in metres, properties for members as for full frame.

o - column boundary node
+ - floor boundary node

Key to edge fixity.

Mode A :- o - fully restrained; + - fully restrained.

B :- o - " " + - Z deflection restrained.

C :- o - Z deflection restrained; + - Z " "

TABLE 6.15 - End Forces for Member 1 - 2 of Sub-frame of Fig. 6.12(c).

Boundary Edge Fixity Mode	END 1			END 2		
	SZ1 (KN)	MX1 (KNM)	MY1 (KNM)	SZ2 (KN)	MX2 (KNM)	MY2 (KNM)
(a) Roof member						
A	41.4	9.7	-25.3	60.0	6.0	77.7
B	48.2	17.2	-29.0	69.1	11.4	90.7
C	43.7	8.1	-18.5	78.8	7.1	115.5
Full frame analysis	51.8	22.1	-41.6	63.6	13.4	77.4
(b) Lower storey member						
A	44.8	13.4	-37.0	59.6	9.9	79.2
B	51.3	21.7	-42.3	68.2	16.4	91.6
Full frame analysis	57.8	26.6	-64.2	60.3	20.9	74.6

For key to boundary edge fixity see Fig 6.12(c).

6.9 Comments on Sub-framing Results.

The possibility of using half the member stiffness for adjacent members, when considering plane sub-frames, is put forward in the Handbook on the Unified Code for structural concrete (CP110:1972)[1]. However, the Joint Committee's Second Report on 'Fully-rigid multi-storey welded steel frames'[2] does not go this far and suggests that actual stiffness for adjacent beam members is sufficient. Since difficulties arise in the use of half stiffness factors when considering the floor deck no modification of adjacent beam stiffness has been employed.

The results for the internal member case are in good agreement with the full frame results, especially when the fixing mode was with the boundary column nodes fully restrained and the floor boundary nodes restrained from vertical movement. The other forms of boundary restraint provide results, as given in TABLE 6.12(a), as expected. For example, for the fully fixed boundary edge condition, load is attracted more to the boundary thus producing lower member end force values.than for the full analysis. Similar agreement of results with that of the full frame are shown in TABLE 6.12(b) for internal members at roof level. It is interesting to note that the mode of fixity, mode B, gives the best results in each case.

The sub-frames for the boundary members have proved difficult to formulate completely, despite the regular nature of the frame. The three types of boundary condition are shown for both roof and lower storey members in Fig. 6.12(a), (b) and (c). The symmetrical cases were quite successful with the same boundary

condition as for the internal members. This is shown in TABLE 6.14 for both roof and lower storey beams. However, the results for the non-symmetrical cases are not so encouraging; the shear and moment values at node 1 in each case are lower than those obtained from the full frame solution. This is evident in TABLES 6.13 and 6.15. The closest agreement is again provided by the same boundary conditions as used in the previous sub-frames. It would appear that the stiffness of joint 1 is not such that it is sufficient to attract load from its surrounding area. The effect of halving the beam stiffness for the adjacent member[1] was investigated by a moment distribution on the plane frame, and, although this does increase the moment value at end 1, it is not enough to produce results near the required value. It can be seen in TABLES 6.13 and 6.15 that several boundary conditions have been tried in order to improve results, but all have fallen short of the acceptable values.

6.10 References.

1. Handbook on the Unified Code for structural concrete. (CP110:1972) - Cement and Concrete Association. p.2.
2. Joint Committee's Second Report on 'Fully-rigid multi-storey welded steel frames'. I.C.E. The Welding Institute. May 1971. p.23-24.

7. DISCUSSION OF DECKED FRAME SOLUTION AND COMPARATIVE TESTING.

7.1 Discussion of the Program and the Method of Solution Described in Section 5.

The basic objective of producing a computer program for the analysis of space frame structures incorporating floor decks was completed; the techniques described in section 5 having been successfully implemented. However, the program cannot be said to be in a finished state since there are further extensions, refinements and improvements which could be made.

The incorporation of floor elements has proved highly successful in allowing versatility in mesh placing and floor layout. The problem of slab/beam continuity has been overcome; however, the in-plane rotational stiffness of the slab still remains undefined since it has only really been bypassed in the program. The use of separate modes, i.e. flexural and shear, for the floor matrices produces a more efficient system and creates less demand for working store. It also enabled the condensation of the floor matrices to be easily implemented. However, despite the gains from this condensation the original floor matrices will invariably be non-banded, meaning that the whole matrix must be held in working store (note: only half the matrix is stored because of symmetry).

Taking advantage of similarity between floors proved successful. However, the present approach could be less restrictive, allowing floors not totally similar in all respects

to be used as similar floors in the program i.e. floor similarity depends on floor loading which could be made an independant variable. This extension would create a more efficient system. There is also a restriction in the type of floor loading which can be applied, but this is not too severe since most of the required forms can be handled.

The condensation of floor matrices reduced the size of working space and improved the efficiency of backing store with regard to data transfer. However, the floor results do not yet provide the in-plane stresses or principal bending stresses which would enhance the output of the program.

The condensation of the structure matrix did not appear as successful as was hoped since its efficiency depends on the size of the structure to be analysed. The 'part-structure matrix', like the floor matrices, are kept in an un-banded form but here it is evident that the matrices, because of the nature of the specified numbering system, would prove banded. However, if the number of columns per floor is low the advantage in using a banded form would be small, although for large systems such a use would reduce the storage requirements and thus improve the success of the 'part-structure matrix' condensation section.

The backing store system used reduced the number of data transfers to a minimum and allowed the use of the program, under the standard operating system on the computer available, to be fairly easy.

The examples given illustrate the variety in the type of structure which can be analysed using the program. This ranges

from general decked frame structures to more specialist structures such as bridge decks and even shear walls. The results given show the considerable amount of information obtained from the program and the various means of illustrating it.

The results from the examples show how the floor slabs interact with the rest of the structure and contribute to the total stiffness of the frame, thus showing that any assumption that the floors act as rigid diaphragms would be in error. Example 1, which is a type of structure that cannot be simulated by a series of plane frames, illustrates how the floor load is distributed between supporting columns, showing, as stated in section 5, that simple assumptions for the calculation of column loading would provide inaccurate results.

7.2 Advantages and Limitations of the DECK1 Program.

In obtaining a complete three-dimensional analysis of a structure, it was inevitable that certain disadvantages and limitations would be created in the developed program.

Basic assumptions about the structure may be in error, such as the structure being rigidly connected. Other disadvantages appear in the program restrictions, for example the confinement to orthogonal structures. Drawbacks occur with all programs. The usual one of system dependence is perhaps the most disappointing, as it prevents the immediate use of the program on other machines, and unfortunately this program is no exception.

The size of the structure which can be analysed is restricted by the size of the store available and software limitations inherent in the system. However, using an Elliot 4130 machine with 64k core store plus backing store, the program can handle multi-storey structures of up to 500 nodes (up to 2000-3000 degrees of freedom).

The data input and output section could be improved, since only a basic format was used initially for the prototype program. However, the implementation of a more sophisticated system, such as that given for the TEE1 program in section 2, is possible. Also the method of input data storage is inefficient, since each member's properties are stored independently, and an improvement would have to be undertaken. The output form is sufficient but could be improved to handle user specification.

7.3 Discussion of Comparative Test Results of Section 6.

It was intended that the results from the comparative testing would provide an insight into the variations in analyses obtained from different program types so that an attempt to simulate three-dimensional behaviour in a less complex analysis could be instigated. Also undertaken was an investigation into the effect of mesh concentration on structure and floor results from which came further information about the interaction between columns and floors. Finally a preliminary investigation into the use of three-dimensional sub-frames for determining beam forces for members of a space frame, without recourse to the full frame solution, was entered upon.

The comparative frame results in Test 1 illustrate the higher bending and in-plane stiffness of the clad frame over the bare skeletal frame. The effect of increasing the beam width in the less complex analyses, in order to account for the floor stiffness, has had limited success but improvements in all the loading cases were achieved by this approach. For the uniform floor loading, even with a beam width of the order of 1/9th of the span of the slab, the results were still some way short of what was required. However, for the sway and twisting load cases, the 700mm beam width does provide reasonable comparative results. In the beam supported case the best comparison of column forces with those from the DECK1 analysis were obtained from the moment distribution of the plane frame assuming a triangular loading pattern on the beam. Here the beam was considered as a tee section with a flange width equal to 1/6th of the span of the slab.

The tests on mesh density to establish its effect on floor and structure solutions did show that, in order to provide floor representation in an analysis, it is not generally necessary to use a high mesh density. It appears from the results given that a relatively coarse mesh is satisfactory because increases in mesh concentration do not have large effects on structure results. However, if detailed information of the floor behaviour is required a high mesh concentration will probably be necessary especially at locations of high stress.

The investigations on the three-dimensional sub-framing were not completely successful, however, some basic trends were

established and a platform for further research provided. Sub-frames for the internal members appear to be satisfactory with respect to the full test frame but the boundary cases have proved more difficult to simulate. Attempts using the recommended half stiffness for adjacent beams did not provide the answer, nor did the various floor edge conditions described. For the sub-frames that were reasonably successful it was established that one floor boundary condition was used in the best solutions.

It is noted that the limited success was achieved with only the symmetric frames, and that the established sub-frames may not be successful for use with irregular frames. However, it is thought that the floor boundary condition established will probably prove satisfactory in any sub-frames used for members in irregular frames.

7.4 Conclusions and Recommendations.

From the foregoing discussion it is clear that where only a simple analysis is to be used, due allowance for the extra stiffness contributed by the slab could be simulated by an adjustment to the beam properties. It would appear that further work in this field would provide sufficient information on which to base recommendations as to how to assess these adjustments. It may be that, instead of changing member section properties it would be better to adjust the actual inertia properties individually.

The mesh concentration tests point to the fact that it is not necessary to have dense mesh patterns in order to achieve representation of the floor stiffness. However, if the floor deformation and forces are required, it may be necessary to have a finer mesh; but since the storage requirements are dependent upon the number of mesh nodes, these should be kept to a minimum, Thus further tests to aid the fixing of mesh concentration requirements for floor stiffness representation would be beneficial.

The sub-frame techniques appear to be feasible but as yet the tests have only provided limited information. The mode for the floor boundary restraint has been recommended and guidance for future investigation is provided. It is expected that three-dimensional sub-frames will be established but further work on these lines is required. With the availability of such sub-frames it may be possible to instigate testing in order to establish beam sections for use in a Tee-beam analysis which simulates space frame behaviour.

8. SUMMARY.

8.1 Summary of Completed Research.

A new approach to the analysis of three-dimensional structures is provided in the DECK1 program. The techniques employed have proved successful and enabled a reduction of storage requirements. The program especially favours multi-storey structures due to its 'part-structure' condensation process.

The handling of the floor slab incorporation is of prime importance. The use of split modes and floor condensation reduces storage demands which are created by high density finite element patterns on the floor.

The comparative tests undertaken have provided a better understanding of three-dimensional behaviour. The bending action of the floor deck is well illustrated together with its subsequent interaction with the whole frame. The tests have provided an insight into the problem of approximating complete structure analysis by a simpler frame one using revised beam sections.

The sub-frame investigation, although not providing full information for the specification of space frame member sub-frames, does provide fixing recommendations for the floor slab boundary. Despite being of a limited nature, the results do give encouragement to the hope that these sub-frames can be fully established.

8.2 Suggestions for Future Work.

The program, as it stands, is not complete but does provide a new approach for the commissioning of a more thorough user-orientated program which would include the improvements and refinements to the techniques employed in DECK1, as explained in earlier sections.

It would now appear that for the final stage of complete structure analysis, the incorporation of wall elements would be required. The present work provides ideas as to how to approach the problem with regard to storage reduction. However, problems such as wall/slab continuity would produce difficulties.

The comparative testing has provided information on how to approximate space frame analysis into a simpler mode whilst accounting for floor action. It shows the limitations and restrictions of the various techniques applied, with special emphasis on the effective beam width. Unfortunately the work is by no means exhaustive and there still remains a large area of investigation to be undertaken. A key to further work may be provided by the use of sub-framing which was also investigated. It is believed that it would be beneficial to establish full specification of three-dimensional sub-frames for space frame members in order that comparative testing on a sub-frame level could be undertaken. From this it may be possible to set up recommended effective beam widths for assessing the extra stiffness of a floor clad space frame by means of a plane frame analysis.

COMPUTATIONAL ANALYSIS OF MULTI-STOREY FRAMED DECK STRUCTURES

APPENDIX

Algol listing of DECK1 program.

A program for the elastic analysis of orthogonal space frame structures incorporating floor slabs.

September 1974.

A brief summary of the basic program parameters is presented below. The input data is given in the required sequence for the DECK1 program. The 'Program call name' is the program identifier where the specified data is stored.

Program input quantities.	Program call name.
1). Elastic moduli.	
Young's modulus.	E
Shear modulus.	G
2). Formal parameters.	
Number of structure joints.	JOIN
" " " members.	MEMB
" " floors.	FLOR
Maximum number of structure nodes per floor.	NODE
Number of floor types.	TYPES

Number of structure restraints. REST .

3). Floor parameters.

Maximum number of elements per floor. ELM

" " " restraints per floor. MAXLINK

" " " mesh points per floor. MESH

4). Structure coordinates - one set for each structure joint.

X - coordinate. XC

Y - " . YC

Z - " . ZC

5). Member properties for beams and columns -

one set for each member.

Member end node I. CON

" " " J. CON

Area of cross-section. PAR

Inertia about Y-axis. I_y PAR

" " Z-axis. I_z PAR

Polar moment of inertia. J PAR

6). Structure restraints - one set for each restraint.

Restrained node. RST

Mode of restraint. RST

7). Floor type specification.

Floor type numbers listed in ascending sequence

of floors. TYPE

Nos. 8 to 14 refer to a particular floor type and are repeated for each type of floor.

8). Specify floor type. I .

9). Floor type parameters.

Number of mesh points on floor. EPF

Number of structure nodes on floor. EPF

" " elements on floor. EPF

" " restraints on floor. EPF

10). Floor properties.

Young's modulus. PROP

Poisson's Ratio. PROP

Thickness of floor. PROP

11). Ordered list of structure/floor connections. (cf. No. 15)

Listed for structure node ascending order -

mesh points coincident with structure nodes. IP

12). Floor restraints - one set for each restraint.

Restrained mesh point. LINK

Mode of restraint. LINK

13). Element connection list - one set for each row of elements.

Elements are listed in rows taken in the X direction on the floor.

Number of elements in row.

Mesh points of first element in row

-- i, j, k, l. SCON

14). Floor mesh coordinates - one set for each mesh point.

X - coordinate. XS

Y - coordinate. YS

15). Ordered list of structure/floor connections. (cf. No. 11)

Listed for structure node ascending order -

one set for each floor.

Structure nodes coincident with mesh points. IPR

The loading data input is defined as follows:-

16). Uniformly distributed loading on floors.

UDL listed in type ascending order for all floor types.

17). Floor shear loading - one set for each floor type.

a). Floor type.

b). Loading - one set for each load.

Mesh point at which load is applied.

Value of load.

Type of load and its direction of action.

18). Structure loading - one set for each floor.

a). Floor number.

b). Loading - one set for each load.

Joint at which load is applied.

Value of load.

Type of load and its direction of action.

End of input data definition.

Notes on data input.

1. Structure node numbering must follow a sequential system from floor to floor. i.e. one floor must be totally numbered before moving on to the next.

2. Floor mesh numbering is assumed to be fixed for X-direction numbering. If Y-direction numbering is undertaken then for 13. above elements will be listed in rows in the Y-direction.

3. None of the formal parameters in 2. should be zero. If the value is actually zero use a dummmy value - 1. This should only happen for those variables connected with the floors - where individual values are defined later in the input.

ALGOL 4100

```
 1    DECK1;
 2    "COMMENT"  STIFFNESS MATRIX ANALYSIS OF A SPACE FRAME
 3       INCORPORATING A FINITE ELEMENT ANALYSIS OF FLOOR DECKS;
 4    "BEGIN""INTEGER" JOIN,MEMB,FLOR,NODE,TYPES,REST,
 5        MESH,
 6                     I,J,K,P,S,ELM,MAXLINK;
 7            "REAL"  E,G;
 8         "READ"     E,G;
 9    "PRINT"'/F';
10    "PRINT" '/L','      SPACE  FRAME  ANALYSIS  --  ELASTIC
11                  INCORPORATING FLOOR DECKS
12                      AUGUST  1972';
13    DIGITS(6);
14        ALIGNED(6,3);
15    "PRINT"'/L2'                INPUT  DATA'L2';
16    "PRINT"'/L',SAMELINE,'YOUNGS MODULUS  ',E,'   SHEAR MODULUS  ',G;
17         "READ"  JOIN,MEMB,FLOR,NODE,TYPES,REST,ELM,MAXLINK;
18     "READ" MESH;
19    "PRINT"'/L',SAMELINE,'PARAMETERS';
20    "PRINT"'/L',SAMELINE,' JOINTS  ',JOIN;
21    "PRINT"'/L',SAMELINE,'  MEMBERS  ',MEMB;
22    "PRINT"'/L',SAMELINE,  'FLOORS  ',FLOR;
23    "PRINT"'/L',SAMELINE,  '  RESTRAINTS  ',REST;
24    "PRINT"'/L',' FLOOR  PARAMETERS';
25    "PRINT"'/L',SAMELINE,' MAX NODES PER FLOOR ',NODE;
26    "PRINT"'/L',SAMELINE, 'NUMBER OF FLOOR TYPES ',TYPES;
27    "PRINT"'/L',SAMELINE,ELM,' ELEMENTS';
28    "PRINT"'/L',SAMELINE,'MESH= ',MESH,' MAXLINK= ',MAXLINK;
29         "BEGIN""REAL""ARRAY" XC,YC,ZC[1:JOIN],DEF[1:6*JOIN],
30        SHLD[1:TYPES,1:3*MESH],
31         PAR[1:MEMB,1:4],PROP[1:TYPES,1:3],
32           XS,YS[1:TYPES,1:MESH];
33                "INTEGER""ARRAY"  CON[1:MEMB,1:2],RST[1:REST,1:2],
34                                  SCON[1:ELM,1:TYPES,1:4],
35          IPR[1:FLOR,1:NODE],     EPF[1:TYPES,1:4],LINK[1:TYPES,1:MAXLINK,
36     1:2],IP[1:TYPES,1:NODE],     TYPE[1:FLOR];
37         "PROCEDURE" TRANS(A,AT,N,M);
38         "VALUE" N,M; "INTEGER" N,M; "REAL""ARRAY" A,AT;
39         "BEGIN""INTEGER" I,J;
40         "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
```

```
41          "FOR" J:=1 "STEP" 1 "UNTIL" M "DO"  AT[J,I]:=A[I,J];
42          "END" OF TRANS;
43          "PROCEDURE" MATMULT(A,B,C,N,P,Q);
44          "VALUE" N,P,Q; "INTEGER" N,P,Q; "REAL""ARRAY" A,B,C;
45          "BEGIN""INTEGER" I,J,K;
46          "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
47          "FOR" J:=1 "STEP" 1 "UNTIL" Q "DO"
48          "BEGIN"  C[I,J]:=0;
49          "FOR" K:=1 "STEP" 1 "UNTIL" P "DO"
50              C[I,J]:=C[I,J]+A[I,K]*B[K,J];
51          "END";
52          "END" OF MATMULT;
53          "PROCEDURE" MEMBER(I,MM,CON,PIN,XC,YC,ZC,PAR);
54          "VALUE" I; "INTEGER" I; "INTEGER""ARRAY" CON,PIN;
55          "REAL""ARRAY" MM,XC,YC,ZC,PAR;
56      "BEGIN""INTEGER"JJ,P;"REAL" IY,IZ,J,AR,DX,DY,DZ,L;
57                  "REAL""ARRAY" K[1:6,1:6],C,A[1:12,1:6],AT[1:6,1:12];
58           JJ:="IF" CON[I,1]<CON[I,2]"THEN"CON[I,1]"ELSE"CON[I,2];
59           P:="IF" CON[I,1]>CON[I,2]"THEN" CON[I,1]"ELSE"CON[I,2];
60          DX:=XC[JJ]-XC[P];DY:=YC[JJ]-YC[P]; DZ:=ZC[JJ]-ZC[P];
61           L:=SQRT(DX*DX+DY*DY+DZ*DZ);
62      AR:=PAR[I,1];IY:=PAR[I,2];IZ:=PAR[I,3];J:=PAR[I,4];
63          "FOR" JJ:=1 "STEP" 1 "UNTIL" 6 "DO"
64          "FOR" P:=1  "STEP" 1 "UNTIL" 6 "DO"
65          "BEGIN"  A[JJ,P]:="IF" JJ=P "THEN" 1 "ELSE" 0;   K[JJ,P]:=0;
66                   A[JJ+6,P]:="IF" JJ=P "THEN" -1 "ELSE" 0; "END";
67           K[1,1]:=G*J/L;
68           K[2,2]:=E*IY/L;
69           K[3,3]:=E*IZ/L;
70        K[4,4]:=E*AR/L;
71           K[5,5]:=12*K[3,3]/L/L;
72           K[6,6]:=12*K[2,2]/L/L;
73           A[2,6]:=A[8,6]:=-L/2;
74          A[3,5]:=A[9,5]:=L/2;
75            TRANS(A,AT,12,6);
76            MATMULT(A,K,C,12,6,6);
77            MATMULT(C,AT,MM,12,6,12);
78      EXIT: "END" OF MEMBER;
79            "PROCEDURE" TRANSFORM(MM,CON,XC,YC,ZC,I);
80          "VALUE" I;"INTEGER" I;"REAL""ARRAY" MM,XC,YC,ZC;
81            "INTEGER""ARRAY" CON;
82          "BEGIN""INTEGER" J,II,JJ,KK,P;"REAL""ARRAY" C,LA,K,LAT[1:3,1:3];
83          "FOR" J:=1,2,3 "DO"
84          "FOR" P:=1,2,3 "DO"  LA[J,P]:=LAT[J,P]:=0;
85        II:="IF"CON[I,1]<CON[I,2]"THEN"CON[I,1]"ELSE"CON[I,2];
86        JJ:="IF"CON[I,1]>CON[I,2]"THEN"CON[I,1]"ELSE"CON[I,2];
87            "IF" ABS(XC[II]-XC[JJ])>0,0001 "THEN""GOTO" EX;
88            "IF" ABS(YC[II]-YC[JJ])>0,0001 "THEN""GOTO" WHY;
89            "IF" ABS(ZC[II]-ZC[JJ])>0,0001 "THEN""GOTO" ZED;
90            "GOTO" ERROR;
91      EX:"IF"XC[JJ]>XC[II]"THEN"
92      "BEGIN"LA[1,1]:=LA[2,2]:=LA[3,3]:=1;"END"
93      "ELSE""BEGIN"LA[1,1]:=LA[2,2]:=-1;LA[3,3]:=1;    "END";
94      "GOTO" AGAIN;
95      WHY: "IF"YC[JJ]>YC[II]"THEN"
96      "BEGIN"LA[1,2]:=-1;LA[2,1]:=LA[3,3]:=1; "END"
97      "ELSE""BEGIN"LA[1,2]:=LA[2,1]:=-1;LA[3,3]:=1; "END";
98      "GOTO" AGAIN;
99      ZED: "IF"ZC[JJ]>ZC[II]"THEN"
100     "BEGIN"LA[1,3]:=-1;LA[2,2]:=LA[3,1]:=1; "END"
```

```
101   "ELSE""BEGIN"LA[1,3]:=LA[2,2]:=1;LA[3,1]:=-1; "END";
102   AGAIN:  TRANS(LA,LAT,3,3);
            "FOR" J:=0,3,6,9 "DO"
            "FOR"KK:=0,3,6,9 "DO" "BEGIN"
            "FOR" II:=1,2,3 "DO"
            "FOR" JJ:=1,2,3 "DO"  K[II,JJ]:=MM[II+J,JJ+KK];
               MATMULT(LA,K,C,3,3,3);
               MATMULT(C,LAT,K,3,3,3);
            "FOR" II:=1,2,3  "DO"
            "FOR" JJ:=1,2,3  "DO"  MM[II+J,JJ+KK]:=K[II,JJ]; "END";
            "GOTO" EXIT;
103   ERROR:"PRINT"('(L)',SAMELINE,'ERROR IN COORDINATES');
104   EXIT:"END" OF TRANSFORM;
105   "PROCEDURE" FINITE(A,B,D,P,C);
106   "VALUE" A,B,D,P; "REAL" A,B,D,P; "ARRAY" C;
107   "BEGIN""INTEGER" I,J; "REAL" PP,F,S,T; "ARRAY" K[1:12,1:12];
            "PROCEDURE" MATADD(A,B,C,N);
            "VALUE" N; "INTEGER" N;  "ARRAY" A,B,C;
            "BEGIN""INTEGER" I,J;
            "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
            "FOR" J:=1 "STEP" 1 "UNTIL" N "DO"
              A[I,J]:=B[I,J]+C[I,J];
            "END" OF MATADD;
108   "FOR" I:=1 "STEP" 1 "UNTIL" 12 "DO"
109   "FOR" J:=1 "STEP" 1 "UNTIL" 12 "DO" C[I,J]:=K[I,J]:=0;
110     S:=A/B;  T:=(1-P)/2;     F:=D/15/A/B;
            I:=0; J:=0;
111   "FOR"PP:=60,0,0,30,0,20,30,0,15,60,0,0,0,0,0,15,0,10,30,0,20,-60,0,
      -30,-30,0,-15,60,0,0,0,0,0,0,0,0,30,0,10,15,0,5,-30,0,20,-30,0,-15,
      -60,0,-30,30,0,-15,60,0,0,0,0,0,0,0,0,0,0,0,15,0,5,30,0,10,-15,0,10,
      -30,0,20) "DO""BEGIN" J:=J+1;
            "IF" J>I "THEN""BEGIN" I:=I+1; J:=1;"END";
                 C[I,J]:=PP/S/S; "END";
            I:=0; J:=0;
112   "FOR"PP:=60,-30,20,0,0,0,-60,30,0,60,-30,10,0,30,20,0,0,0,0,0,0,30,-15
      ,0,-30,-15,0,60,-15,10,0,15,5,0,-30,20,0,0,0,0,0,0,0,0,0,-30,15,0,30,
      15,0,-60,30,0,60,-15,5,0,15,10,0,-30,10,0,30,20,0,0,0,0,0,0,0,0,0,0,
      0,0,0 "DO""BEGIN"  J:=J+1;
            "IF" J>I "THEN""BEGIN" I:=I+1; J:=1; "END";
                 K[I,J]:=PP*S*S; "END";
             MATADD(C,K,C,12);
            I:=0; J:=0;
113   "FOR"PP:=30,-15,0,15,-15,0,-30,0,-15,30,0,0,0,15,0,-15,0,0,15,15,0,
      -30,15,0,30,0,0,30,15,0,0,0,0,0,-15,0,0,0,0,0,0,0,-15,15,0,30,0,0,-30,
      -15,0,-30,0,15,30,0,0,0,-15,0,0,0,0,0,15,0,0,0,0,0,0,0,15,0,0,-15,
      -15,0  "DO""BEGIN"  J:=J+1;
            "IF" J>I "THEN""BEGIN" I:=I+1; J:=1; "END";
                 K[I,J]:=PP*P;   "END";
             MATADD(C,K,C,12);
            I:=0; J:=0;
114   "FOR"PP:=84,-6,8,6,0,8,-84,6,-6,84,-6,-2,0,6,8,-6,0,-8,6,0,8,-84,6,-6,
      84,6,6,84,6,-8,0,-6,2,0,-6,8,6,0,-2,-6,0,2,-6,0,8,84,-6,6,-84,-6,-6,
      -84,6,6,84,6,2,0,-6,-8,0,-6,-2,0,6,8,-6,0,2,6,0,-2,6,0,-8,-6,0,8
            "DO""BEGIN"    J:=J+1;
            "IF" J>I "THEN""BEGIN" I:=I+1; J:=1; "END";
                 K[I,J]:=PP*T;   "END";
             MATADD(C,K,C,12);
115   "FOR" I:=2,5,8,11 "DO"
116   "FOR" J:=1 "STEP" 1 "UNTIL" 12 "DO""BEGIN"
117   C[I,J]:=C[I,J]*B;    C[I+1,J]:=C[I-1,J]*A;"END";
```

```
161   "FOR" I:=1  "STEP" 1 "UNTIL" 12 "DO"
162   "FOR" J:=2,5,8,11 "DO""BEGIN"
163   C[I,J]:=C[I,J]*B;  C[I,J+1]:=C[I,J+1]*A; "END";
164        "FOR" J:=1 "STEP" 1 "UNTIL" 12 "DO"
165        "FOR" I:=J "STEP" 1 "UNTIL" 12 "DO" "BEGIN"
166         C[I,J]:=F*C[I,J]; C[J,I]:=C[I,J]; "END";
167   "COMMENT" MATRIX TRANSFORMATION TO STRUCTURE COORDINATES;
168   "FOR"I:=1,4,7,10 "DO"
169   "FOR" J:=1"STEP"1"UNTIL"12"DO"
170   "BEGIN" C[I,J]:=-C[I,J];
171          C[J,I]:=-C[J,I];          "END";
172   "END" OF PROCEDURE FINITE;
173   "PROCEDURE" ASSEMBLE(II,JJ,KK,LL,NN,C,B);
174   "VALUE" II,JJ,KK,LL,NN; "INTEGER" II,JJ,KK,LL,NN; "ARRAY" C,B;
175   "BEGIN""INTEGER" I,J,R1,R2,R3,R4,T,I1,J1,K,P;
176    R1:=3*(II-1); R2:=3*(JJ-1); R3:=3*(KK-1); R4:=3*(LL-1);
177        "FOR" K:=R1,R2,R3,R4 "DO"
178        "FOR" I:=1 "STEP" 1 "UNTIL" 3 "DO"
179        "FOR" J:=I "STEP" 1 "UNTIL" 3 "DO" "BEGIN"
180         I1:=K+I; J1:=K+J; T:=(2*NN-I1)*(I1-1)/2+J1;
181        "IF" K=R1 "THEN" P:=0;
182        "IF" K=R2 "THEN" P:=6;
183        "IF" K=R3 "THEN" P:=3;
184        "IF" K=R4 "THEN" P:=9;
185             C[T]:=C[T]+B[I+P,J+P]; "END";
186        "FOR" I:=1 "STEP" 1 "UNTIL" 3 "DO"
187        "FOR" J:=1 "STEP" 1 "UNTIL" 3 "DO""BEGIN"
188         I1:=R1+I;
189        "FOR" K:=R2,R3,R4 "DO""BEGIN" J1:=K+J;
190   "IF" R1>K "THEN""BEGIN" I1:=K+I; J1:=R1+J;"END";
191         T:=(2*NN-I1)*(I1-1)/2+J1;
192        "IF" K=R2 "THEN" P:=6;
193        "IF" K=R3 "THEN" P:=3;
194        "IF" K=R4 "THEN" P:=9;
195   "IF" R1>K "THEN""BEGIN" C[T]:=C[T]+B[I+P,J];
196   "GOTO" M1;"END";
197   C[T]:=C[T]+B[I,J+P];
198   M1:    "END";
199         I1:=R2+I;
200        "FOR" K:=R3,R4 "DO""BEGIN" J1:=K+J;
201   "IF" R2>K "THEN""BEGIN" I1:=K+I;J1:=R2+J;"END";
202         T:=(2*NN-I1)*(I1-1)/2+J1;
203        "IF" K=R3 "THEN" P:=3;
204        "IF" K=R4 "THEN" P:=9;
205   "IF" R2>K "THEN""BEGIN" C[T]:=C[T]+B[I+P,J+6];
206   "GOTO" M2; "END";
207     C[T]:=C[T]+B[I+6,J+P];
208   M2:  "END";
209   I1:=R3+I;          J1:=R4+J;
210   "IF" R3>R4 "THEN""BEGIN" I1:=R4+I;J1:=R3+J;"END";
211   T:=(2*NN-I1)*(I1-1)"DIV"2+J1;
212   "IF" R3>R4 "THEN""BEGIN" C[T]:=C[T]+B[I+9,J+3];
213   "GOTO" M3;  "END";
214         C[T]:=C[T]+B[I+3,J+9];
215   M3:    "END";
216        "END" OF  ASSEMBLE;
217        "PROCEDURE" FLIM(B,C,P,N);
218        "VALUE" N,P; "INTEGER" N,P; "ARRAY" B,C;
219        "BEGIN""INTEGER" I,J,K,KK,T1,T2,T3;"REAL" PIV;
220        I:=0;
```

```
221   R1:   I:=I+1;
222         "IF" I>P "THEN""GOTO" EXIT;
223         T1:=(N+N-I)*(I-1)"DIV"2+I;
224         PIV:=B[T1];
225         PIV:=1.0/SQRT(PIV);
226         B[T1]:=B[T1]*PIV;
227         C[I]:=C[I]*PIV;
228         J:=I;
229   R2:   J:=J+1;
230         "IF" J>N "THEN""GOTO" R1;
231         T1:=(N+N-I)*(I-1)"DIV"2+J;
232         "IF" B[T1]=0 "THEN""GOTO" R2;
233         B[T1]:=B[T1]*PIV;
234        C[J]:=C[J]-B[T1]*C[I];
235         K:=I;
236   R3:   K:=K+1;
237         "IF" K>J "THEN""GOTO" R2;
238         T2:=(N+N-K)*(K-1)"DIV"2+J;
239         T3:=(N+N-I)*(I-1)"DIV"2+K;
240         B[T2]:=B[T2]-B[T1]*B[T3];
241         "GOTO" R3;
242   EXIT:"END" OF ELIM;
243         "PROCEDURE" MILE(B,X,Y,P,N);
244         "VALUE" N,P; "INTEGER" N,P;  "ARRAY" B,X,Y;
245         "BEGIN""INTEGER" I,J,K,PP,T; "REAL" U;
246          PP:=P;
247         "IF" P=N "THEN""BEGIN"
248       T:=N*(N+1)/2;
249          X[N]:=Y[N]/B[T];
250          PP:=P-1;
251         "END";
252         "FOR" I:=PP "STEP" -1 "UNTIL" 1 "DO"
253         "BEGIN" U:=0;
254         "FOR" J:=I+1 "STEP" 1 "UNTIL" N"DO"
255         "BEGIN"  T:=(2*N-I)*(I-1)/2+J;
256                  U:=U+B[T]*X[J]; "END";
257   T:=(2*N-I)*(I-1)/2+I;
258          X[I]:=(Y[I]-U)/B[T];
259         "END";
260         "END" OF MILE;
261   "PROCEDURE" TODISC(CH,A,SEC,II,N);
262   "VALUE" CH,II,N; "INTEGER" CH,N,II,SEC; "ARRAY" A;
263   "BEGIN""INTEGER" I,J,DUMMY; "ARRAY" B[1:64];
264         SEC:=DCHECK(CH); J:=0;
265   "FOR" I:=II "STEP" 1 "UNTIL" N "DO""BEGIN" J:=J+1; B[J]:=A[I];
266   "IF" I=N "THEN""BEGIN" DWRITE(CH,B,1,J); DUMMY:=DCHECK(CH);
267                          "GOTO" OUT; "END";
268   "IF" J<64 "THEN""GOTO" OUT;
269         DWRITE(CH,B,1,64); DUMMY:=DCHECK(CH); J:=0;
270   OUT: "END";
271   "END" OF TODISC;
272   "PROCEDURE" FROMDISC(CH,A,SEC,II,N);
273   "VALUE" CH,SEC,II,N; "INTEGER" CH,SEC,II,N; "ARRAY" A;
274   "BEGIN""INTEGER" I,J,DUMMY;   "ARRAY" B[1:64];
275         DFIND(CH,SEC);  J:=II-1;
276   RET: DREAD(CH,B,1,64);   DUMMY:=DCHECK(CH);
277         "FOR" I:=1 "STEP" 1 "UNTIL" 64 "DO"
278   "BEGIN"     J:=J+1;
279   "IF"J>N"THEN""GOTO"EXIT;
280         A[J]:=B[I];
```

```
281         "END";
282         "IF" L<N "THEN""GOTO" RET;
283    EXIT:
284    "END" OF FRONDISC;
285          "PROCEDURE" FORMLOAD(LL,U,I,J,K,L,A,B);
286    "VALUE"U,I,J,K,L,A,B;"INTEGER"I,J,K,L;"REAL"A,B,U;"ARRAY"LL;
287         "BEGIN""REAL" LOAD; "INTEGER" T,TT;
288               LOAD:=U*A*B;
289         "FOR" T:=I,J,K,L "DO""BEGIN"
290               TT:=(T-1)*3+1;
291         LL[TT]:=LL[TT]+LOAD/4;"END";
292         "END" OF FORMLOAD;
293         "PROCEDURE" ORDER(ROW,P,TY);
294         "VALUE" P,TY; "INTEGER" P,TY; "INTEGER""ARRAY" ROW;
295         "BEGIN""INTEGER" I,J,S;
296          S:=0;
297         "FOR" I:=1 "STEP" 1 "UNTIL" EPF[TY,1] "DO"
298         "BEGIN"
299         "FOR" J:=1 "STEP" 1 "UNTIL" EPF[TY,2] "DO"
300       "BEGIN""IF" I=IP[TY,J] "THEN""GOTO" EXIT; "END";
301          S:=S+1;
302          ROW[I]:=S;
303    EXIT:"END";
304         "FOR" I:=1 "STEP" 1 "UNTIL" EPF[TY,2] "DO"
305         "BEGIN" J:=IP[TY,I];
306                 S:=S+1;
307               ROW[J]:=S;
308         "END";
309         "END" OF ORDER;
310        "PROCEDURE"  INSTORE(FORCE,DEF,FLR,V);
311         "VALUE" FLR,V; "INTEGER" FLR,V;
312    "ARRAY" FORCE,DEF;
313       "BEGIN""INTEGER"  I,J,K,II,T;
314           ALIGNED(6,4);   DIGITS(6);
315    "PRINT"//F\\;
316    "PRINT"//L\\,SAMELINE,'DISPLACEMENTS OF JOINTS ON FLOOR\',FLR-1;
317    "PRINT"//L2\\,'   JOINT'S8'ROTX'S8'ROTY'S8'ROTZ'S8'\\,
318            'DEFX'S8'DEFY'S8'DEFZ'\;
319    "IF" FLR=1"THEN"T:=1"ELSE"T:=FLR-1;
320    "FOR"K:=1"STEP"1"UNTIL"EPF[TYPE[T],2]"DO""BEGIN"
321    "IF" FLR=1 "THEN""BEGIN" I:=K;"GOTO" JUMP;"END";
322    I:=IPR[FLR-1,K];
323    JUMP:
324    "PRINT"//L\\,SAMELINE,I;
325    II:=(I-1)*6;      J:=(K-1)*6;
326         "FOR" II:=1 "STEP" 1 "UNTIL" 6 "DO"
327         "BEGIN" DEF[II+I]:=FORCE[II+J];
328    "PRINT" SAMELINE,FORCE[II+J];
329                 FORCE[II+J]:=0;     "END";
330      "END";
331     "FOR"K:=1"STEP"1"UNTIL" EPF[TYPE[FLR],2] "DO"
332    "BEGIN"
333         "IF" V= 12*NODE "THEN""BEGIN"
334          I:=IPR[FLR,K]; I:=6*(I-1); J:=K+NODE; J:=6*(J-1);
335    "PRINT" //L\\,SAMELINE,IPR[FLR,K];
336         "FOR"II:=1"STEP"1"UNTIL" 6 "DO"
337         "BEGIN" DEF[II+I]:=FORCE[II+J];
338        "PRINT" SAMELINE,FORCE[II+J];
339       FORCE[II+J]:=0;          "END";
340    "GOTO" EXIT;
```

```
341             "END";
342          "FOR" II:=1 "STEP" 1 "UNTIL" 6 "DO""BEGIN"
343           J:=K+NODE;  J:=6*(J-1); FORCE[II+J]:=0;"END";
344    EXIT;"END";
345          "END" OF INSTORE;
346          "PROCEDURE" OUTSTORE(FORCE,DEF,FLR);
347          "VALUE" FLR; "INTEGER" FLR; "ARRAY" FORCE,DEF;
348          "BEGIN""INTEGER" I,J,K,II;
349     "FOR"I:=1"STEP"1"UNTIL"12*NODE"DO" FORCE[I]:=0;
350     "FOR"K:=1"STEP"1"UNTIL" EPF[TYPE[FLR],2] "DO"
351          "BEGIN" I:=IPR[FLR,K];
352                  I:=6*(I-1);
353       J:=K+NODE;
354                  J:=6*(J-1);
355     "FOR"   II:=1,2,3,4,5,6 "DO" FORCE[J+II]:=DEF[I+II];
356          "END";
357          "END" OUTSTORE;
358          "PROCEDURE" ADDREST(RST,FLR,MX,N);
359          "VALUE" FLR,N; "ARRAY" MX; "INTEGER""ARRAY" RST;
360    "INTEGER"    FLR,N;
361    "BEGIN""INTEGER" I,K,II,JJ,T,NN;
362    NN:=12*N;
363          "FOR" I:=1 "STEP" 1 "UNTIL" REST "DO"
364          "BEGIN"  II:=RST[I,1]; JJ:=RST[I,2];
365    "IF" FLR=1"THEN""BEGIN"
366     "FOR"K:=1"STEP"1"UNTIL" EPF[TYPE[FLR],2] "DO"
367      "BEGIN"
368    "IF" II=K "THEN""GOTO" GRD;"END";
369    "GOTO" OUT;
370    GRD: K:=(K-1)*6+JJ;
371    T:=(NN+NN-K)*(K-1)"DIV"2+K;
372    MX[T]:=MX[T]+10,0↑20;
373    "GOTO" EXIT;
374    "END";
375    OUT;
376     "FOR"K:=1"STEP"1"UNTIL" EPF[TYPE[FLR],2] "DO"
377      "BEGIN"
378        "IF"  II=IPR[FLR,K]"THEN""GOTO" FIX;  "END";
379       "GOTO" EXIT;
380    FIX;  K:=(K+NODE-1)*6+JJ;
381    T:=(NN+NN-K)*(K-1)"DIV"2+K;
382          MX[T]:=MX[T]+10,0↑20;
383    EXIT;"END";
384          "END" OF ADDREST;
385          "PROCEDURE" ADDLOAD(LOAD,NODE,FLR);
386          "VALUE" NODE,FLR; "INTEGER" NODE,FLR; "ARRAY" LOAD;
387          "BEGIN""INTEGER" I,J,F,DIR,JT,NLOAD; "REAL" VAL;
388           ALIGNED(6,3); DIGITS(6);
389          "READ" F;
390          "IF" F=FLR"THEN""GOTO" ON "ELSE""GOTO"EXIT;
391    ON:   "READ" NLOAD;
392          "PRINT"'('L'\',SAMELINE,' FLOOR ',F,' WITH ',NLOAD, 'LOADS.');
393     "IF" NLOAD=0 "THEN""GOTO" EXIT;
394          "FOR" I:=1 "STEP" 1 "UNTIL" NLOAD "DO"
395          "BEGIN""READ" JT,VAL,DIR;
396          "PRINT"'('L'\',SAMELINE,'JOINT ',JT,' VALUE ',VAL,
397                        ' DIRECTION ',DIR;
398     "FOR"J:=1"STEP"1"UNTIL" EPF[TYPE[FLR],2] "DO"
399          "BEGIN""IF" JT=IPR[F,J] "THEN""GOTO" OUT; "END";
400          "PRINT"'('L'\',' LOADING ERROR ';
```

```
401         "GOTO" EXIT;
402     OUT:  J:=(NODE*J-1)*6;
403           J:=J+DIR;
404           LOAD[J]:=LOAD[J]+VAL;
405         "END";
406     EXIT:"END" OF ADDLOAD;
407         "PROCEDURE" RAT(DX,DY,AT,L);
408         "VALUE" DX,DY,L; "REAL" DX,DY,L;    "ARRAY" AT;
409         "BEGIN""INTEGER" I,J;
410         "FOR" I:=1 "STEP" 1 "UNTIL" 6 "DO"
411         "FOR" J:=1,2,3 "DO" AT[I,J]:=0;
412       AT[1,1]:=1;    AT[4,1]:=-1;
413       AT[2,1]:=AT[5,1]:=DY/2;
414       AT[3,1]:=AT[6,1]:=-DX/2;
415       AT[2,2]:=AT[3,3]:=DX/L;
416       AT[5,2]:=AT[6,3]:=-DX/L;
417       AT[2,3]:=AT[6,2]:=-DY/L;
418       AT[3,2]:=AT[5,3]:=DY/L;
419         "END" OF RAT;
420         "PROCEDURE" RAT2(DX,DY,AT,L);
421         "VALUE" DX,DY,L; "REAL" DX,DY,L;    "ARRAY" AT;
422         "BEGIN""INTEGER" I,J;
423         "FOR" I:=1 "STEP" 1 "UNTIL" 6 "DO"
424         "FOR" J:=1,2,3 "DO" AT[I,J]:=0;
425       AT[1,1]:=1;    AT[4,1]:=-1;
426       AT[1,3]:=AT[4,3]:=L/2;
427       AT[2,2]:=AT[3,3]:=DX/L;
428       AT[5,2]:=AT[6,3]:=-DX/L;
429       AT[2,3]:=AT[6,2]:=-DY/L;
430       AT[3,2]:=AT[5,3]:=DY/L;
431         "END" OF RAT2;
432         "INTEGER""PROCEDURE" MINO(I,J);
433         "VALUE" I,J;    "INTEGER" I,J;
434         "BEGIN""IF" I<J "THEN" MINO:=I "ELSE" MINO:=J;
435         "END" OF MINO;
436         "INTEGER""PROCEDURE" MAXO(I,J);
437         "VALUE" I,J;    "INTEGER" I,J;
438         "BEGIN""IF" I>J "THEN" MAXO:=I "ELSE" MAXO:=J;
439         "END" OF MAXO;
440         "PROCEDURE" ASCEND(PNT,COD,TY,N);
441         "VALUE" TY,N; "INTEGER" TY,N;
442         "ARRAY" COD;  "INTEGER""ARRAY" PNT;
443         "BEGIN""INTEGER" I,K;                  "INTEGER" PIV;
444     RET:      K:=0;
445         "FOR" I:=1 "STEP" 1 "UNTIL" N-1 "DO"
446         "BEGIN""IF" COD[TY,PNT[I]]>COD[TY,PNT[I+1]] "THEN"
447         "BEGIN"  K:=K+1;  PIV:=PNT[I];  PNT[I]:=PNT[I+1];
448                  PNT[I+1]:=PIV;  "END";
449         "END";
450         "IF" K>0 "THEN""GOTO" RET;
451         "END" OF ASCEND;
452          "PROCEDURE" STIFF(SM,I,J,XS,YS,M,TY,PAR,S,SW);
453         "VALUE" I,J,M,TY,S,SW;    "INTEGER" I,J,M,TY,S,SW;
454         "ARRAY" SM,XS,YS,PAR;
455         "BEGIN""INTEGER" I1,I2,P,Q,N,K;
456         "REAL" DX,DY,L,IN,JN;     "ARRAY" A,AT[1:6,1:3],ST[1:3,1:3];
457          I1:=MINO(I,J);           I2:=MAXO(I,J);
458         DX:=XS[TY,I2]-XS[TY,I1];
459         DY:=YS[TY,I2]-YS[TY,I1];
460         L:=SQRT(DX*DX+DY*DY);
```

```
461         "FOR" P:=1,2,3 "DO"
462         "FOR" Q:=1,2,3 "DO"  ST[P,Q]:=0;
463         "IF" S=0 "THEN""BEGIN"
464         RAT(DX,DY,AT,L);
465         IN:=PAR[M,2]; JN:=PAR[M,4];
466         "IF" SW>0 "THEN""BEGIN" DX:=L; DY:=0; "END";
467         ST[1,1]:=12*E*IN/L/L/L;  ST[2,2]:=G*JN/L; ST[3,3]:=E*IN/L;
468             RAT(DX,DY,A,L);
469         "END""ELSE""BEGIN"
470         RAT2(DX,DY,AT,L);
471         IN:=PAR[M,3];  JN:=PAR[M,1];
472         "IF" SW>0"THEN""BEGIN" DX:=L; DY:=0; "END";
473                 RAT2(DX,DY,A,L);
474      ST[1,1]:=E*IN/L; ST[2,2]:=E*JN/L;  ST[3,3]:=12*E*IN/L/L/L;
475         "END";
476         "FOR" N:=1 "STEP" 1 "UNTIL" 6 "DO"
477         "FOR" K:=1 "STEP" 1 "UNTIL" 6 "DO"
478         "BEGIN"   SM[N,K]:=0;
479         "FOR" P:=1,2,3 "DO"
480     "FOR"Q:=1,2,3"DO" SM[N,K]:=SM[N,K]+A[N,P]*ST[P,Q]*AT[K,Q];
481         "END";
482         "END" OF STIFF;
483         "PROCEDURE" SMASSEM(I,J,SM,B,N);
484         "VALUE" N,I,J; "INTEGER" N,I,J; "ARRAY" SM,B;
485     "BEGIN""INTEGER" I1,J1,K,L,P,Q,S,D,T;
486          I1:=MINO(I,J);  J1:=MAXO(I,J);  S:=0; D:=0;
487           "IF" I1=I "THEN" D:=3"ELSE"S:=3;
488           I1:=(I1-1)*3;       J1:=(J1-1)*3;
489         "FOR" K:=1,2,3 "DO"
490         "FOR" L:=K "STEP" 1 "UNTIL" 3 "DO"
491         "BEGIN" P:=I1+K; Q:=I1+L; T:=(N+N-P)*(P-1)"DIV"2+Q;
492                 B[T]:=B[T]+SM[K+S,L+S];
493                 P:=J1+K; Q:=J1+L; T:=(N+N-P)*(P-1)"DIV"2+Q;
494                 B[T]:=B[T]+SM[K+D,L+D]; "END";
495         "FOR"K:=1,2,3 "DO"
496         "FOR"L:=1,2,3 "DO""BEGIN"
497         P:=I1+K;  Q:=J1+L;
498         T:=(N+N-P)*(P-1)"DIV"2+Q;
499          B[T]:=B[T]+SM[K+S,L+D]; "END";
500         "END" OF SMASSEM;
501         "PROCEDURE" BEEF(I,J,S,SM,FORCE);
502         "VALUE" I,J,S; "INTEGER" I,J,S; "ARRAY" SM,FORCE;
503         "BEGIN""INTEGER" P,Q,II,JJ,A,B;
504                "ARRAY" FOR,DIS[1:6];
505         II:=MINO(I,J); JJ:=MAXO(I,J);       A:=II; B:=JJ;
506         II:=3*(II-1);  JJ:=3*(JJ-1);
507         "FOR" P:=1 "STEP" 1 "UNTIL" 6 "DO" FOR[P]:=DIS[P]:=0;
508         "FOR" P:=1,2,3 "DO""BEGIN"
509              DIS[P]:=FORCE[II+P];
510              DIS[P+3]:=FORCE[JJ+P];  "END";
511         "FOR" P:=1 "STEP" 1 "UNTIL" 6 "DO"
512         "FOR" Q:=1 "STEP" 1 "UNTIL" 6 "DO"
513              FOR[P]:=FOR[P]+SM[P,Q]*DIS[Q];
514     "PRINT"'/L'',   ALIGNED(6,3);  DIGITS(6);
515     "PRINT"'/L'',SAMELINE,A,B,''S6'';
516         "FOR" P:=1 "STEP" 1 "UNTIL" 6 "DO""PRINT" SAMELINE,FOR[P];
517         "END";
518         "PROCEDURE" MESHGEN(TY,IS,IF,P,Q,MEM,S,FLORMX,ROW,SW);
519         "VALUE" TY,IS,IF,MEM,S,SW; "INTEGER" TY,IS,IF,MEM,S,SW;
520     "INTEGER""ARRAY" ROW; "ARRAY" P,Q,FLORMX;
```

```
521         "BEGIN""INTEGER" I,J,K,M,N;
522         "INTEGER""ARRAY" PNT[1:MESH"DIV"3]; "ARRAY" SM[1:6,1:6];
523          J:=0;
524     "FOR"I:=1"STEP"1"UNTIL" EPF[TY,1] "DO"
525         "BEGIN""IF" Q[TY,IS]"NE"Q[TY,I] "THEN""GOTO" NOMESH;
526               "IF" I=IS "OR" I=IF "THEN""GOTO" NOMESH;
527               "IF" P[TY,I]<P[TY,IS]"THEN" "GOTO" NOMESH;
528               "IF" P[TY,I]>P[TY,IF]"THEN" "GOTO" NOMESH;
529            J:=J+1;    PNT[J]:=I;
530     NOMESH;    "END";
531         N:=J;
532               ASCEND(PNT,P,TY,N);
533         M:=0; I:=IS;
534     RET;     M:=M+1;
535         "IF" M>N "THEN""BEGIN" J:=IF; "GOTO" FIX; "END";
536         J:=PNT[M];
537     FIX;      STIFF(SM,I,J,XS,YS,MEM,TY,PAR,S,SW);
538       "IF" S=0"THEN" SMASSEM(ROW[I],ROW[J],SM,FLORMX,3*EPF[TY,1])
539         "ELSE"  BEEF(I,J,S,SM,FLORMX);
540               I:=J;
541         "IF" M<N+1 "THEN""GOTO" RET;
542         "END" OF MESHGEN;
543         "PROCEDURE" ADDBEAM(FT,N,E,G,S,FLORMX,ROW,SS);
544         "VALUE" FT,N,E,G,S,SS; "INTEGER" FT,N,S,SS; "REAL" E,G;
545         "ARRAY" FLORMX; "INTEGER""ARRAY" ROW;
546         "BEGIN""INTEGER" I,J,K,M,IS,IF,FLR,TY,SW;
547         "IF" SS>0 "THEN""BEGIN"  FLR:=FT;  TY:=TYPE[FLR]; "END"
548                   "ELSE""BEGIN""FOR" I:=1 "STEP" 1 "UNTIL" FLOR "DO"
549                         "BEGIN""IF" TYPE[I]=FT "THEN" FLR:=I;
550                                    TY:=FT;   "END";    "END";
551         "IF" SS=100 "THEN" SW:=0 "ELSE" SW:=SS;
552         "IF" SW=1 "THEN""BEGIN"
553     "PRINT"'/'F'';
554         "IF" S=0 "THEN""BEGIN""PRINT"'/'L'',
555          '   BEAM ELEMENT FORCES - FLEXURAL: FLOOR ',SAMELINE,FT;
556     "PRINT"'/'L'',SAMELINE,'ELEMENT NODES'S10'SZ1'S8'MX1'S8'MY1',
557               '/'S8'SZ2'S8'MX2'S8'MY2'; "END""ELSE"
558         "BEGIN""PRINT"'/'L'',
559     '     BEAM ELEMENT FORCES - SHEAR:    FLOOR ',SAMELINE,FT;
560         "PRINT"'/'L'',SAMELINE,'ELEMENT NODES'S10'MZ1'S8'AX1'S8'SY1',
561               '/'S8'MZ2'S8'AX2'S8'SY2'; "END";
562         "END";
563         "FOR" M:=1 "STEP" 1 "UNTIL" MEMB "DO"
564         "BEGIN"         I:=CON[M,1];         J:=CON[M,2];
565         "IF" ABS(ZC[I]-ZC[J])>0.0001 "THEN""GOTO" NOBEAM;
566         "IF" ZC[I] "NE" ZC[IPR[FLR,1]] "THEN""GOTO" NOFLOR;
567      "FOR"K:=1"STEP"1"UNTIL" EPF[TY,2] "DO"
568         "BEGIN""IF" I=IPR[FLR,K] "THEN" IS:=IP[TY,K];
569               "IF" J=IPR[FLR,K] "THEN" IF:=IP[TY,K]; "END";
570         "IF" ABS(XS[TY,IS]-XS[TY,IF])<0.0001 "THEN""BEGIN"
571         "IF" YS[TY,IS]>YS[TY,IF] "THEN""BEGIN" K:=IS;IS:=IF;IF:=K;
572         "END";MESHGEN(TY,IS,IF,YS,XS,M,S,FLORMX,ROW,SW);"END""ELSE"
573         "BEGIN""IF" XS[TY,IS]>XS[TY,IF] "THEN""BEGIN"K:=IS;IS:=IF;
574         IF:=K; "END";  MESHGEN(TY,IS,IF,XS,YS,M,S,FLORMX,ROW,SW);
575         "END";
576     NOFLOR;
577     NOBEAM;
578         "END";
579         "END" OF ADDBEAM;
580         "PROCEDURE" COLUMN(DEF,CON,XC,YC,ZC,PAR);
```

```
      "INTEGER""ARRAY" CON; "ARRAY" DEF,XC,YC,ZC,PAR;
      "BEGIN""INTEGER" I,J,M;  "INTEGER""ARRAY" PIN[1:1];
      "PROCEDURE" FORCES(I,CON,PIN,XC,YC,ZC,LOAD,PAR);
      "VALUE" I; "INTEGER" I;  "INTEGER""ARRAY" CON,PIN;
      "REAL""ARRAY" XC,YC,ZC,LOAD,PAR;
      "BEGIN""INTEGER" J,KK,II,JJ;"REAL""ARRAY" LA,LAT[1:3,1:3],
                  MM[1:12,1:12],
                     C[1:3,1:1],   XVECT,FVECT[1:12,1:1],VECT[1:3,1:1];
      "FOR" J:=1,2,3 "DO"
"FOR" KK:=1,2,3"DO" LA[J,KK]:=LAT[J,KK]:=0;
       JJ:="IF"CON[I,1]<CON[I,2]"THEN"CON[I,1]"ELSE"CON[I,2];
       KK:="IF"CON[I,1]>CON[I,2]"THEN"CON[I,1]"ELSE"CON[I,2];
   "IF" ABS(XC[JJ]-XC[KK])>0,0001 "THEN""GOTO" EX;
      "IF" ABS(YC[JJ]-YC[KK])>0,0001 "THEN""GOTO" WHY;
      "IF" ABS(ZC[JJ]-ZC[KK])>0,0001 "THEN""GOTO" ZED;
EX; "IF" XC[KK]>XC[JJ] "THEN"
"BEGIN"LA[1,1]:=LA[2,2]:=LA[3,3]:=1;"END"
"ELSE""BEGIN"LA[1,1]:=LA[2,2]:=-1;LA[3,3]:=1;    "END";
"GOTO" AGAIN;
WHY; "IF"YC[KK]>YC[JJ] "THEN"
"BEGIN"LA[1,2]:=-1;LA[2,1]:=LA[3,3]:=1; "END"
"ELSE""BEGIN"LA[1,2]:=LA[2,1]:=-1;LA[3,3]:=1; "END";
"GOTO" AGAIN;
ZED; "IF" ZC[KK]>ZC[JJ] "THEN"
"BEGIN"LA[1,3]:=-1;LA[2,2]:=LA[3,1]:=1; "END"
"ELSE""BEGIN"LA[1,3]:=LA[2,2]:=1;LA[3,1]:=-1; "END";
AGAIN;   TRANS(LA,LAT,3,3);
       JJ:=(JJ-1)*6;  KK:=(KK-1)*6;
       MEMBER(I,MM,CON,PIN,XC,YC,ZC,PAR);
      "FOR" J:=1 "STEP" 1 "UNTIL" 6 "DO" "BEGIN"
       XVECT[J,1]:= LOAD[J+JJ]; XVECT[J+6,1]:=LOAD[J+KK];"END";
      "FOR" KK:=0,3,6,9 "DO" "BEGIN"
     "FOR" J:=1,2,3  "DO"   VECT[J,1]:=XVECT[J+KK,1];
           MATMULT(LAT,VECT,C,3,3,1);
       "FOR" J:=1,2,3 "DO"XVECT[J+KK,1]:=C[J,1]; "END";
           MATMULT(MM,XVECT,FVECT,12,12,1);
      DIGITS(3);    ALIGNED(5,2);
      "PRINT"''L'',SAMELINE,CON[I,1],'=',CON[I,2],''S4'';
      "FOR" J:=1 "STEP" 1 "UNTIL" 12 "DO"
      "PRINT"SAMELINE,FVECT[J,1]; "END";
      "PRINT" ''F'';
      "PRINT" ''L'',SAMELINE,'  COLUMN FORCES';
      "PRINT" ''L'','MEMBER'S10'MX1'S6'MY1'S6'MZ1'S6'AX1'S6'SY1',
      ''S6'SZ1'S6'MX2'S6'MY2'S6'MZ2'S6'AX2'S6'SY2'S6'SZ2';
      "FOR" M:=1 "STEP" 1 "UNTIL" MEMB "DO"
      "BEGIN"  I:=CON[M,1];    J:=CON[M,2];
      "IF" ABS(ZC[I]-ZC[J])>0,001 "THEN""GOTO" COL
                                  "ELSE""GOTO" EXIT;
COL:      FORCES(M,CON,PIN,XC,YC,ZC,DEF,PAR);
EXIT:"END";
      "END" OF COLUMN;
      "PROCEDURE" SHEARLOAD(LD,ROW,TY,S,SH);
      "VALUE" TY,S;  "INTEGER" TY,S;
      "ARRAY" LD,SH; "INTEGER""ARRAY" ROW;
      "BEGIN""INTEGER" NOLOAD,JT,DIR,I,II,TP;  "REAL" VAL;
       ALIGNED(5,3); DIGITS(6);
      "IF" S=1 "THEN""GOTO" SHEAR;
      "FOR" I:=1 "STEP" 1 "UNTIL" 3*MESH "DO" SH[TY,I]:=0;
      "READ" TP; "PRINT"''L'',SAMELINE,' TYPE ',TP;
      "IF" TY"NE"TP "THEN""BEGIN""PRINT"''L'',
```

```
641             ' WRONG TYPE; LOADING SET TO ZERO.';
642         "GOTO" EXIT; "END";
643         "READ" NOLOAD;
644         "PRINT"'''L'',SAMELINE,'SHEAR LOADING; ',NOLOAD,' LOADS.';
645         "IF" NOLOAD=0 "THEN""GOTO" EXIT;
646         "PRINT"'''L'','JOINT      VALUE      DIRECTION';
647         "FOR" I:=1 "STEP" 1 "UNTIL" NOLOAD "DO"
648         "BEGIN""READ"  JT,VAL,DIR;
649         "PRINT"'''L'',SAMELINE,JT,''S2'',VAL,''S4'',DIR;
650          II:=3*(JT-1); II:=II*DIR;
651          SHCTY,II]:=SHCTY,II]*VAL;
652         "END";
653     SHEAR;   "FOR" I:=1"STEP"1"UNTIL" EPFCTY,1] "DO"
654     "BEGIN" "INTEGER" J,K,L;
655     "FOR" J:=1,2,3 "DO""BEGIN"
656        K:=ROWCI];
657       K:=3*(K-1)+J;
658        L:=3*(I-1)+J;
659        LDCK]:=SHCTY,C];
660     "END";
661     "END";
662     EXIT;      "END" OF SHEARLOAD;
663         "PROCEDURE" STRESS(P,E,T,A,B,M,FAC);
664         "VALUE" P,E,T,A,B,FAC;  "REAL" P,E,T,A,B,FAC;
665         "REAL""ARRAY" M;
666         "BEGIN""REAL" R,S,V,U,W,X,Y,Z,P1,P2,F,K1,K2;
667         "INTEGER" I,J;
668     "PROCEDURE" TWIST(ES);
669     "ARRAY" ES;
670         "BEGIN""ARRAY" ATC1:3,1:3]; "REAL" W;
671     "INTEGER" I,J;
672         "FOR"I:=1,2,3 "DO"
673         "FOR"J:=1,2,3 "DO""BEGIN"
674         W:=ESCI,J+3]; ESCI,J+3]:=ESCI,J+6]; ESCI,J+6]:=W;
675         W:=ESCI+3,J+9];ESCI+3,J+9]:=ESCI+6,J+9]; ESCI+6,J+9]:=W;
676         "END";
677         "FOR" I:=1,2,3 "DO"
678         "FOR" J:=I "STEP" 1 "UNTIL" 3 "DO""BEGIN"
679         W:=ESCI+6,J+6]; ESCI+6,J+6]:=ESCI+3,J+3]; ESCI+3,J+3]:=W;
680         "END";
681         "FOR" I:=1,2,3 "DO""FOR"J:=1,2,3"DO" ATCI,J]:=ESCI+3,J+6];
682         "FOR" I:=1,2,3 "DO""FOR"J:=1,2,3"DO" ESCI+3,J+6]:=ATCJ,I];
683     "END"OF TWIST;
684         "FOR"I:=1"STEP"1"UNTIL"12"DO"
685         "FOR"J:=1"STEP"1"UNTIL"12"DO" MCI,J]:=0;
686          P1:=(1.0-P)/2;    P2:=(1.0+P)/2;
687          R:=4*(B*B*P1*A*A);       S:=4*(A*A+P1*B*B);
688          V:=3*A*B*P2;             X:=3*A*B*(P1-P);
689          U:=2*(P1*A*A-2*B*B);     W:=2*(B*B-2*P1*A*A);
690          Y:=2*(A*A-2*P1*B*B);     Z:=2*(P1*B*B-2*A*A);
691          F:=(E*T)/(12*A*B*(1.0-P*P));
692          K1:=E*T*T*T*FAC*B/12/F/A;
693          K2:=E*T*T*T*FAC*A/12/F/B;
694         "FOR" I:=1,4,7,10 "DO""BEGIN" MCI,I]:=4*(K1+K2);
695             MCI+1,I+1]:=R;    MCI+2,I+2]:=S;     "END";
696         MC4,1]:=MC10,7]:=2*K1;   MC7,1]:=MC10,4]:=2*K2;
697         MC11,2]:=MC8,5]:=-R/2;   MC12,3]:=MC9,6]:=-S/2;
698         MC8,2]:=MC11,5]:=W;      MC5,2]:=MC11,8]:=U;
699         MC6,3]:=MC12,9]:=Y;      MC9,3]:=MC12,6]:=Z;
700         MC3,2]:=MC8,6]:=MC9,5]:=MC12,11]:=V;
```

```
701         M[6,5]:=M[9,8]:=M[11,3]:=M[12,2]:=-V;
702         M[5,3]:=M[9,2]:=M[11,6]:=M[12,8]:=X;
703         M[6,2]:=M[8,3]:=M[11,9]:=M[12,5]:=-X;
704         "FOR" I:=1 "STEP" 1 "UNTIL" 12 "DO"
705         "FOR" J:=I "STEP" 1 "UNTIL" 12 "DO""BEGIN"
706          M[J,I]:=M[J,I]*F;        M[I,J]:=M[J,I];      "END";
707         TWIST(M);
708      "FOR" I:=1"STEP" 1"UNTIL"12"DO"
709      "FOR" J:=1"STEP"1"UNTIL"12"DO""BEGIN"
710        M[J,I]:=M[I,J];  "END";
711         "END" OF STRESS;
712    "PROCEDURE" FLOFIX(TY,M,LINK,Q,N,ROW,FLORMX);
713    "VALUE" TY,M,Q,N; "INTEGER" TY,M,Q,N;
714    "ARRAY" FLORMX;  "INTEGER""ARRAY" LINK,ROW;
715    "BEGIN""INTEGER" I,J,T,JT,JJ,D;
716     "IF"M=0"THEN""GOTO" OUT;
717    "FOR" I:=1"STEP"1"UNTIL"M"DO""BEGIN"
718      JT:=LINK[TY,I,1];  D:=LINK[TY,I,2];
719    "IF" Q=0 "THEN""BEGIN"
720      "IF" D>3 "THEN""GOTO" EXIT; "END";
721    "IF" Q=1 "THEN""BEGIN"
722      "IF" D<3 "THEN""GOTO" EXIT; "END";
723      "IF" D>3 "THEN" D:=D-3;
724         J:=ROW[JT];
725       JJ:=3*(J-1)*D;
726      T:=(N*N-JJ)*(JJ-1)"DIV"2+JJ;
727         FLORMX[T]:=10.0*20;
728    EXIT:   "END";
729      OUT:
730      "END"  OF FLOFIX;
731    "PRINT"//L\\,' STRUCTURE COORDINATES\;
732    "PRINT"//L\\,'     XC            YC                ZC\;
733    "FOR" I:=1"STEP" 1"UNTIL"JOIN"DO""BEGIN"
734    "READ" XC[I],YC[I],ZC[I];
735    "PRINT"//L\\,SAMELINE,XC[I],YC[I],ZC[I];        "END";
736    "PRINT"//L\\,'MEMBER   AREA   IY   IZ     J\;
737    "FOR"I:=1"STEP"1"UNTIL"MEMB"DO"
738    "BEGIN"
739    "READ" CON[I,1],CON[I,2];
740    "PRINT"//L\\,SAMELINE,CON[I,1],CON[I,2];
741    "FOR"J:=1,2,3,4 "DO""READ" PAR[I,J];
742    "FOR"J:=1,2,3,4"DO""PRINT" SAMELINE,PAR[I,J];
743    "END";
744    "PRINT"//L\\,'    RESTRAINTS,\;
745    "PRINT"//L\\,'   NODE    VARIABLE\;
746    "FOR" I:=1"STEP"1"UNTIL"REST"DO""BEGIN"
747    "READ" RST[I,1],RST[I,2]; "PRINT" //L\\,SAMELINE,RST[I,1],RST[I,2];
748    "END";
749    "FOR"I:=1"STEP"1"UNTIL" FLOR "DO""READ" TYPE[I];
750    RET1: "READ" I,EPF[I,1],EPF[I,2],EPF[I,3],EPF[I,4];
751    "PRINT"//L\\,'TYPE MESH NODES ELEMS LINKS\;
752    "PRINT"//L\\,SAMELINE,I,EPF[I,1],EPF[I,2],EPF[I,3],EPF[I,4];
753    "READ" PROP[I,1],PROP[I,2],PROP[I,3];
754    "PRINT"//L\\,'     YM                P.R.                TH,\;
755    "PRINT"//L\\,SAMELINE,PROP[I,1],PROP[I,2],PROP[I,3];
756        "FOR"J:=1"STEP"1"UNTIL" EPF[I,2] "DO""READ" IP[I,J];
757       "PRINT"  //L\\,'    FLOOR/COL    NODES\;
758    "PRINT" //L\\;
759    "FOR"J:=1"STEP"1"UNTIL"EPF[I,2]"DO""PRINT" SAMELINE,IP[I,J];
760     "IF"EPF[I,4]=0"THEN""GOTO" AVOID;
```

```
761     "PRINT" //L\\,'      FLOOR   RESTRAINTS\;
762         "FOR"J:=1"STEP"1"UNTIL" EPF[I,4] "DO""READ" LINK[I,J,1],
763                                                  LINK[I,J,2];
764      "FOR"J:=1"STEP"1"UNTIL"EPF[I,4]"DO"
765     "PRINT"//L\\,SAMELINE,LINK[I,J,1],LINK[I,J,2];
766      AVOID;
767         "BEGIN""INTEGER" A,B,C,D;
768     "PRINT"//L\\,' ELS          1ST ELEMENT OF ROW\;
769          S:=0;
770     RET2:"READ" P,A,B,C,D;
771     "PRINT"//L\\,SAMELINE,P,' == ',A,B,C,D;
772         "FOR" K:=0 "STEP" 1 "UNTIL" P-1 "DO"
773         "BEGIN" S:=S+1;
774          SCON[S,I,1]:=A*K;  SCON[S,I,2]:=B*K;
775          SCON[S,I,3]:=C*K;  SCON[S,I,4]:=D*K;
776         "END";
777         "IF" S<EPF[I,3] "THEN""GOTO" RET2;
778         "END";
779     "PRINT"//L\\,'  FLOOR MESH COORDINATES\;
780     "PRINT"//L\\,'     XS              YS\;
781     "FOR"J:=1"STEP"1"UNTIL" EPF[I,1] "DO""BEGIN"
782     "READ" XS[I,J],YS[I,J]; "PRINT"//L\\,SAMELINE,XS[I,J],YS[I,J];
783     "END";
784         "IF" I<TYPES "THEN""GOTO" RET1;
785       "PRINT"  //L\\,'   STRUCTURE/FLOOR  NODES\;
786         "FOR" I:=1 "STEP" 1 "UNTIL" FLOR "DO"
787      "BEGIN""PRINT" //L\\;
788      "FOR" J:=1"STEP"1"UNTIL" EPF[TYPE[I],2]"DO""READ" IPR[I,J];
789      "FOR" J:=1"STEP"1"UNTIL"EPF[TYPE[I],2]"DO"
790     "PRINT" SAMELINE,IPR[I,J]; "END";
791         "COMMENT" FLOOR CONDENSATION SECTION;
792         "BEGIN""INTEGER" NN,MM,KK,FLR,TY,CH,DOC,DC;
793     "INTEGER""ARRAY" SECF,SECS[1:2*TYPES],SECN[1:2*FLOR];
794         "REAL""ARRAY" LOAD[1:TYPES];
795     "PRINT"//F\\;
796     "PRINT"//L2\\,'       LOADING  DATA.\,//L\\;
797          ALIGNED(6,6);
798       "FOR" TY:=1"STEP"1"UNTIL" TYPES "DO"
799         "BEGIN"
800              "READ" LOAD[TY];
801              "PRINT"//L\\,SAMELINE,//S8\\,'FLOOR TYPE ',TY,
802                           ' UDL ',LOAD[TY];
803         "END";
804         "FOR" CH:=25,26,27,28,29 "DO"  DOPEN(CH,0,0);
805          NN:=12*NODE;
806          NN:=NN*(NN+1)"DIV"2;
807          TY:=0;
808     RET3: TY:=TY+1;
809          MM:=3*EPF[TY,1]; P:=MM;
810          KK:=MM*(MM+1)"DIV"2;
811          DOC:=3*EPF[TY,2];
812          DOC:=P-DOC;
813     DC:=(P*P-DOC)*(DOC=1)"DIV"2+P;
814         "BEGIN""REAL""ARRAY" FLORMX[1:KK],FLORLD[1:P];
815         "INTEGER""ARRAY" ROW[1:EPF[TY,1]];
816                "INTEGER" YY;
817         "FOR" I:=1 "STEP" 1 "UNTIL" KK "DO" FLORMX[I]:=0;
818         "FOR" I:=1 "STEP" 1 "UNTIL" EPF[TY,1] "DO" ROW[I]:=0;
819         "FOR" I:=1 "STEP" 1 "UNTIL" P "DO" FLORLD[I]:=0;
820              ORDER(ROW,P,TY);
```

```
821         "BEGIN""INTEGER" K,L,EL;
822     "REAL" A,B,D,PP,AA,BB,U,YM,T;"REAL""ARRAY" ES[1:12,1:12];
823           AA:=BB:=0;
824           U:=LOAD[TY];
825         "FOR" EL:=1 "STEP" 1 "UNTIL" EPF[TY,3] "DO"
826         "BEGIN"  I:=SCON[EL,TY,1];  J:=SCON[EL,TY,2];
827                  K:=SCON[EL,TY,3];  L:=SCON[EL,TY,4];
828     A:=XS[TY,J]-XS[TY,I];          B:=YS[TY,I]-YS[TY,K];
829           A:=ABS(A);             B:=ABS(B);
830     YM:=PROP[TY,1];   PP:=PROP[TY,2];    T:=PROP[TY,3];
831         D:=(YM*T*T)/(12*(1-PP*PP));
832         "IF" AA"NE"A "OR" BB"NE"B "THEN""BEGIN"
833                        FINITE(A,B,D,PP,ES);
834                        AA:=A; BB:=B; "END";
835       I:=ROW[I];
836       J:=ROW[J];
837           K:=ROW[K];
838       L:=ROW[L];
839         ASSEMBLE(I,J,K,L,P,FLORMX,ES);
840      FORMLOAD(FLORLD, U,I,J,K,L,A,B);
841         "END";
842     ADDBEAM(TY,12*NODE,E,G,0,FLORMX,ROW,0);
843      FLOFIX(TY,EPF[TY,4],LINK,0,3*EPF[TY,1],ROW,FLORMX);
844               ELIM(FLORMX,FLORLD,DOC,P);
845           CH:=25;
846           YY:=2*TY-1;
847               TODISC(CH,FLORMX,SECF[YY],DC*1,KK);
848               TODISC(CH,FLORLD,SECF[YY+1],DOC*1,P);
849     "END";
850         "FOR" I:=1 "STEP" 1 "UNTIL" KK "DO" FLORMX[I]:=0;
851         "FOR" I:=1 "STEP" 1 "UNTIL" P  "DO" FLORLD[I]:=0;
852         "BEGIN""INTEGER" K,L,EL;
853         "REAL" A,B,D,PP,AA,BB,T,FAC,YM;
854         "REAL""ARRAY" ES[1:12,1:12];
855           AA:=BB:=0;
856         "FOR" EL:=1 "STEP" 1 "UNTIL" EPF[TY,3] "DO"
857         "BEGIN" I:=SCON[EL,TY,1];  J:=SCON[EL,TY,2];
858         K:=SCON[EL,TY,3];   L:=SCON[EL,TY,4];
859                  A:=XS[TY,J]-XS[TY,I]; B:=YS[TY,I]-YS[TY,K];
860                  A:=ABS(A);        B:=ABS(B);
861           YM:=PROP[TY,1];  PP:=PROP[TY,2];  T:=PROP[TY,3];
862           FAC:=0.25;
863         "IF" AA"NE"A "OR" BB"NE"B "THEN""BEGIN"
864         STRESS(PP,YM,T,A,B,ES,FAC);
865               AA:=A; BB:=B;
866     "END";
867         I:=ROW[I];  J:=ROW[J]; K:=ROW[K];  L:=ROW[L];
868               ASSEMBLE(I,J,K,L,P,FLORMX,ES);
869         "END";
870               SHEARLOAD(FLORLD,ROW,TY,0,SHLD);
871     ADDBEAM(TY,12*NODE,E,G,1,FLORMX,ROW,0);
872      FLOFIX(TY,EPF[TY,4],LINK,1,3*EPF[TY,1],ROW,FLORMX);
873           ELIM(FLORMX,FLORLD,DOC,P);
874           CH:=25;
875           YY:=2*TY-1;
876               TODISC(CH,FLORMX,SECS[YY],DC*1,KK);
877               TODISC(CH,FLORLD,SECS[YY+1],DOC*1,P);
878         "END";
879     "END";
880         "IF" TY<TYPES "THEN""GOTO" RET3;
```

```
881         "BEGIN""REAL""ARRAY" NODEMX[1:NN],NODELD[1:12*NODE],
882                                                 FORCE[1:12*NODE];
883       "INTEGER" V,VV,EX,FD;
884     "PROCEDURE" ADDCOL(FLR,TY,N,E,G);
885     "VALUE" FLR,TY,N,E,G;  "INTEGER" FLR,TY,N; "REAL" E,G;
886     "BEGIN""INTEGER" I,J,K,II,JJ,M,L;  "REAL" LY;
887         "FOR" K:=1 "STEP" 1 "UNTIL" MEMB "DO""BEGIN"
888          I:=CON[K,1];  J:=CON[K,2];
889     LY:=ZC[J]-ZC[I];
890         "IF" ABS(LY)<0.00001 "THEN""GOTO" NOCOL;
891     "IF" ZC[J]>ZC[I] "THEN""BEGIN" JJ:=J; II:=I; "END"
892                           "ELSE""BEGIN" JJ:=I; II:=J; "END";
893      "FOR"L:=1"STEP"1"UNTIL" EPF[TYPE[FLR],2]"DO"
894         "BEGIN""IF" JJ=IPR[FLR,L]"THEN""GOTO" FILL; "END";
895         "GOTO" NOCOL;
896     FILL:"BEGIN""REAL""ARRAY" MM[1:12,1:12],PIN[1:1];
897                  "INTEGER" P,Q,T,I1,J1,V,W;
898               MEMBER(K,MM,CON,PIN,XC,YC,ZC,PAR);
899               TRANSFORM(MM,CON,XC,YC,ZC,K);
900               L:=NODE*L;
901         "IF" FLR=1 "THEN""BEGIN" M:=I; "GOTO" NEXT; "END";
902     "FOR"M:=1"STEP"1"UNTIL" EPF[TYPE[FLR-1],2]"DO"
903         "BEGIN""IF" II=IPR[FLR-1,M] "THEN""GOTO" NEXT; "END";
904     NEXT: M:=(M-1)*6;   L:=(L-1)*6;
905         "IF" II=I "THEN""BEGIN" W:=6; V:=0; "END"
906                   "ELSE""BEGIN" W:=0; V:=6; "END";
907         "FOR" I1:=1 "STEP" 1 "UNTIL" 6 "DO""BEGIN"
908         "FOR" J1:=I1"STEP" 1 "UNTIL" 6 "DO""BEGIN"
909          P:=I1+M; Q:=J1+M;  T:=(N*N-P)*(P-1)"DIV"2+Q;
910              NODEMX[T]:=NODEMX[T]+MM[I1+V,J1+V];
911          P:=I1+L; Q:=J1+L;  T:=(N*N-P)*(P-1)"DIV"2+Q;
912              NODEMX[T]:=NODEMX[T]+MM[I1+W,J1+W]; "END";
913         "FOR" J1:=1 "STEP" 1 "UNTIL" 6 "DO""BEGIN"
914          P:=I1+M; Q:=J1+L;  T:=(N*N-P)*(P-1)"DIV"2+Q;
915              NODEMX[T]:=NODEMX[T]+MM[I1+V,J1+W]; "END";
916         "END";"END";
917     NOCOL:"END";
918         "END" OF ADDCOL;
919         "PROCEDURE" ADJUST(NODE,DOC,NN,P);
920         "VALUE" NODE,DOC,NN,P;  "INTEGER" NODE,DOC,NN,P;
921         "BEGIN""INTEGER" I,J,T1,T2;
922         "FOR" I:=1 "STEP" 1 "UNTIL" DOC "DO"
923         "FOR" J:=I "STEP" 1 "UNTIL" DOC"DO"
924         "BEGIN"
925          T1:=(NN*NN-I)*(I-1)"DIV"2+J;
926          T2:=(NN*NN-I-DOC)*(I+DOC-1)"DIV"2+J+DOC;
927          NODEMX[T1]:=NODEMX[T2];
928          NODEMX[T2]:=0;
929         "END";
930         "FOR" I:=1 "STEP" 1 "UNTIL" DOC "DO"
931         "FOR" J:=DOC+1 "STEP" 1 "UNTIL" P "DO"
932         "BEGIN"  T1:=(NN*NN-I)*(I-1)"DIV"2+J;
933              NODEMX[T1]:=0;  "END";
934         "FOR" I:=1 "STEP" 1 "UNTIL" DOC "DO"
935         "BEGIN"  NODELD[I]:=NODELD[DOC+I];
936                  NODELD[DOC+I]:=0;
937         "END";
938         "END" OF ADJUST;
939       "PROCEDURE" MATSET(BX,N,FLR,EPF);
940     "VALUE" N,FLR;  "INTEGER" N,FLR;
```

```
941    "ARRAY" BX;  "INTEGER""ARRAY" EPF;
942    "BEGIN""INTEGER" I,E,T;
943     E:=6*EPF[TYPE[FLR],2];
944     "IF"FLR=1"THEN""BEGIN"
945    "IF" E=N"THEN""GOTO" OUT;
946    "FOR"I:=N+1"STEP"1"UNTIL"N"DO"
947    "BEGIN" T:=(2*N+2*N-I)*(I=1)"DIV"2+I;
948        BX[T]:=10↑20;
949     "END";
950      OUT;  "END";
951     "IF"E=N"THEN""GOTO" EXIT;
952     "FOR"I:=E+N+1"STEP"1"UNTIL"2*N"DO"
953      "BEGIN" T:=(2*N+2*N-I)*(I-1)"DIV"2+I;
954      BX[T]:=10↑20; "END";
955      EXIT;  "END"  OF  MATSET;
956         "FOR" I:=1 "STEP" 1 "UNTIL" NN "DO" NODEMX[I]:=0;
957         "FOR" I:=1 "STEP" 1 "UNTIL" 12*NODE "DO" NODELD[I]:=FORCE[I]:=0;
958          FLR:=0;
959    RET4; FLR:=FLR+1;
960          TY:=TYPE[FLR];
961          P:=3*EPF[TY,1];  DOC:=3*EPF[TY,2];
962    P:=DOC;
963    MM:=DOC;
964    KK:=MM*(MM+1)"DIV"2;
965         "BEGIN""REAL""ARRAY" BUFMX[1:KK],BUFLD[1:P];
966         "INTEGER" YY;
967    "PROCEDURE" FLASSEM(NODE,FLR,KK,P,NN,SS);
968    "VALUE" NODE,FLR,KK,P,NN,SS;"INTEGER" NODE,FLR,KK,P,NN,SS;
969         "BEGIN""INTEGER" I,J,II,JJ,I1,I2,J1,J2,R,Q,L,M,S,T,W,T1,T2,PP;
970                 "INTEGER""ARRAY"  C[1:3];
971    "IF"  SS=0  "THEN""BEGIN"
972          C[1]:=6;  C[2]:=1;  C[3]:=2;
973    "END""ELSE""BEGIN"
974       C[1]:=3; C[2]:=4; C[3]:=5; "END";
975    T:=S:=NODE;           PP:=P"DIV"3;
976    T:=NODE-1;    S:=NODE;
977         "FOR" I:=1 "STEP" 1 "UNTIL" PP "DO""BEGIN"  S:=S+1;
978    T:=NODE-1+I;
979         "FOR" J:=I "STEP" 1 "UNTIL" PP "DO""BEGIN"  T:=T+1;
980          I1:=(I-1)*3;    I2:=(S-1)*6;
981          J1:=(J-1)*3;    J2:=(T-1)*6;
982         "FOR" II:=1 "STEP" 1 "UNTIL" 3 "DO"
983    "BEGIN""INTEGER"     RR;
984         "IF" I=J "THEN" W:=II "ELSE" W:=1;
985      L:=II+I1;    R:=C[II];
986         "FOR" JJ:=W "STEP" 1 "UNTIL" 3 "DO"
987    "BEGIN" Q:=C[JJ];    R:=C[II];
988    "IF" I=J "THEN""BEGIN"
989    "IF" C[II]=6 "THEN""BEGIN"
990      Q:=C[II];    R:=C[JJ];  "END";
991    "END";
992      Q:=Q+J2;     R:=R+I2;    M:=JJ+J1;
993          T1:=(NN+NN-R)*(R-1)"DIV"2+Q;
994    T2:=(P+P-L)*(L-1)"DIV"2+M;
995          NODEMX[T1]:=BUFMX[T2];
996         "END";
997         RR:=I2+C[II];
998          NODELD[RR]:=BUFLD[L];
999         "END";
1000        "END";"END";
```

```
      "END" OF FLASSEM;
      "FOR" I:=1 "STEP" 1 "UNTIL" KK "DO" BUFMX[I]:=0;
      "FOR" I:=1 "STEP" 1 "UNTIL" P  "DO" BUFLD[I]:=0;
       YY:=2*TY-1;
       FROMDISC(25,BUFMX,SECF[YY],1,KK);
       FROMDISC(25,BUFLD,SECF[YY+1],1,P);
     FLASSEM(NODE,FLR,KK,P,12*NODE,0);
      FROMDISC(25,BUFMX,SECS[YY],1,KK);
     FROMDISC(25,BUFLD,SECS[YY+1],1,P);
      FLASSEM(NODE,FLR,KK,P,12*NODE,1);
      "END";
        ADDCOL(FLR,TY,12*NODE,E,G);
    ADDRAST(RST,FLR,NODEMX,NODE);
ADDLOAD(NODELD,NODE, FLR);
        MATSET(NODEMX,6*NODE,FLR,EPF);
      "IF" FLR=FLOR "THEN""BEGIN" V:=12*NODE; "GOTO" OUT; "END";
       ELIM(NODEMX,NODELD,6*NODE,12*NODE);
RE:    CH:=CH+1;
      "IF" CH=30 "THEN""BEGIN" CH:=25; "GOTO" RE; "END";
       VV:=2*FLR-1;
  ED:=6*BPF[TYPE[FLR],2];
  EX:=(12*NODE+12*NODE-ED)*(ED-1)"DIV"2+12*NODE;
   TODISC(CH,NODEMX,SECN[VV],1,EX);
  TODISC(CH,NODELD,SECN[VV+1],1,ED);
     ADJUST(NODE,6*NODE,12*NODE,12*NODE);
     "GOTO" RET4;
OUT:  ELIM(NODEMX,NODELD,V,12*NODE);
NEW:  MILE(NODEMX,FORCE,NODELD,V,12*NODE);
       INSTORE(FORCE,DEF,FLR,V);
      FLR:=FLR-1;
      "IF" FLR=0 "THEN""GOTO" FINISH;
       VV:=2*FLR-1;
  ED:=6*BPF[TYPE[FLR],2];
  EX:=(12*NODE+12*NODE-ED)*(ED-1)"DIV"2+12*NODE;
  FROMDISC(CH,NODEMX,SECN[VV],1,EX);
  FROMDISC(CH,NODELD,SECN[VV+1],1,ED);
      CH:=CH-1;
      "IF" CH=25 "THEN" CH:=29;
       OUTSTORE(FORCE,DEF,FLR);
  V:=ED;
     "GOTO" NEW;
FINISH:   "END";
      "COMMENT" FLOOR DEFLECTIONS;
      "BEGIN""INTEGER" MM,FLR,TY,KK;
      "PROCEDURE" LAMBDA(A,B,D,P,MS);
      "VALUE" A,B,D,P; "REAL" A,B,D,P;  "ARRAY" MS;
      "BEGIN""INTEGER" I,J;    "REAL" S,T,F,U;
      "FOR" I:=1 "STEP" 1 "UNTIL" 12 "DO"
      "FOR" J:=1 "STEP" 1 "UNTIL" 12 "DO"  MS[I,J]:=0;
           S:=A/B;   U:=B/A;   T:=(1-P)/2;   F:=D/A/B;
      "FOR" I:=1,4,7,10 "DO""BEGIN"
       MS[I,I]:=6*(U+S*P);  MS[I+1,I]:=6*(S+U*P);  "END";
      "FOR" I:=3,6,9,12 "DO""BEGIN"
      "FOR" J:=1,10 "DO"  MS[I,J]:=-2*T;
      "FOR" J:=4,7  "DO"  MS[I,J]:=2*T;    "END";
      MS[1,2]:=MS[7,8]:=-4*A*P;  MS[1,3]:=MS[4,6]:=4*B;
      MS[2,2]:=MS[8,8]:=-4*A;    MS[2,3]:=MS[5,6]:=4*B*P;
      MS[3,2]:=MS[6,5]:=MS[9,2]:=MS[12,5]:=2*B*T;
      MS[3,4]:=MS[5,1]:=MS[8,10]:=MS[11,7]:=-6*S;
      MS[1,4]:=MS[4,1]:=MS[7,10]:=MS[10,7]:=-6*S*P;
```

```
1061      MS[1,7]:=MS[4,10]:=MS[7,1]:=MS[10,4]:=-6*U;
1062      MS[2,7]:=MS[5,10]:=MS[8,1]:=MS[11,4]:=-6*U*P;
1063        MS[3,3]:=MS[6,3]:=MS[9,9]:=MS[12,9]:=-2*A*T;
1064        MS[3,6]:=MS[6,6]:=MS[9,12]:=MS[12,12]:=2*A*T;
1065        MS[3,8]:=MS[6,11]:=MS[9,8]:=MS[12,11]:=-2*B*T;
1066        MS[4,5]:=MS[10,11]:=4*A*P;   MS[8,9]:=MS[11,12]:=-4*B*P;
1067        MS[5,5]:=MS[11,11]:=4*A;     MS[7,9]:=MS[10,12]:=-4*B;
1068    MS[1,5]:=MS[7,11]:=-2*A*P;   MS[2,5]:=MS[8,11]:=-2*A;
1069        MS[1,9]:=MS[4,12]:=2*B;      MS[2,9]:=MS[5,12]:=2*B*P;
1070        MS[4,2]:=MS[10,8]:=2*A*P;    MS[5,2]:=MS[11,8]:=2*A;
1071    MS[7,3]:=MS[10,6]:=-2*B;    MS[8,3]:=MS[11,6]:=-2*B*P;
1072        "FOR" I:=1 "STEP" 1 "UNTIL" 12 "DO"
1073        "FOR" J:=1 "STEP" 1 "UNTIL" 12 "DO"
1074             MS[I,J]:=F*MS[I,J];
1075        "END" OF LAMBDA;
1076       "PROCEDURE" BETA(I,J,K,L,A,B,D,P,FORCE);
1077       "VALUE" I,J,K,L,A,B,D,P; "REAL" A,B,D,P;
1078       "INTEGER" I,J,K,L; "ARRAY" FORCE;
1079        "BEGIN""INTEGER" S,T;     "ARRAY" V,M[1:12],MS[1:12,1:12];
1080         T:=0;                    LAMBDA(A,B,D,P,MS);
1081        "FOR" S:=I,J,K,L "DO""BEGIN"
1082       V[T+1]:=-FORCE[(S-1)*3+1];
1083             V[T+2]:=FORCE[(S-1)*3+2];
1084             V[T+3]:=FORCE[S*3];       T:=T+3;  "END";
1085        "FOR" S:=1 "STEP" 1 "UNTIL" 12 "DO""BEGIN" M[S]:=0;
1086        "FOR" T:=1 "STEP" 1 "UNTIL" 12 "DO" M[S]:=M[S]+MS[S,T]*V[T];
1087    "END";
1088        ALIGNED(5,3);  DIGITS(6);
1089        "PRINT"'/'L';
1090             T:=0;
1091        "FOR" S:=I,J,K,L "DO""BEGIN"
1092        "PRINT" '/'L',SAMELINE,S,M[T+1],M[T+2],M[T+3];
1093                       T:=T+3;
1094        "END";
1095        "END" OF BETA;
1096        "PROCEDURE" GAMMA(FORCE,FLR,XS,YS,SCON,PROP,EPF);
1097        "VALUE" FLR;"INTEGER"FLR;"INTEGER""ARRAY" SCON;
1098    "REAL""ARRAY" FORCE,XS,YS,PROP;
1099    "INTEGER""ARRAY" EPF;
1100    "BEGIN""INTEGER" I,J,K,L,TY,EL;       "REAL"  A,B,D,P,T;
1101    "PRINT"'/'F';
1102    "PRINT"'/'L',SAMELINE,'/S16'SLAB INTERNAL MOMENTS FLOOR',FLR;
1103        ALIGNED(5,3);  DIGITS(6);
1104        "PRINT"'/'L',SAMELINE,'/S1'JOINT,'/S4'MX'S8'MY'S8'MXY';
1105         TY:=TYPE[FLR];
1106        "FOR" EL:=1 "STEP" 1 "UNTIL" EPF[TY,3] "DO"
1107        "BEGIN"  I:=SCON[EL,TY,1];   K:=SCON[EL,TY,3];
1108                 J:=SCON[EL,TY,2];   L:=SCON[EL,TY,4];
1109                 A:=XS[TY,J]-XS[TY,I];
1110          B:=YS[TY,I]-YS[TY,K];
1111             A:=ABS(A);     B:=ABS(B);
1112    T:=PROP[TY,3];
1113             D:=PROP[TY,1];P:=PROP[TY,2];
1114       D:=(D*T*T)/(12*(1-P*P));
1115          BETA(I,K,J,L,A,B,D,P,FORCE);
1116        "END";
1117        "END" OF GAMMA;
1118        "PROCEDURE" RESIN2(DEF,MX,LD,FLR,B);
1119        "VALUE" FLR,B; "INTEGER" FLR,B; "ARRAY" DEF,MX,LD;
1120        "BEGIN""INTEGER" I,J,TY,II,K,L,S,T,L1,L2,L3;
```

```
1121      TY:=TYPE[FLR];    T:=3*EPF[TY,1];
1122         "FOR" K:=1 "STEP" 1 "UNTIL" EPF[TY,2] "DO"
1123         "BEGIN"  II:=IP[TY,K];    I:=IPR[FLR,K];
1124                  I:=(I-1)*6;      II:=(II-1)*3;    S:=0;
1125         "IF" B=0 "THEN""BEGIN" L1:=6; L2:=1; L3:=2; "END"
1126         "ELSE""BEGIN"  L1:=3; L2:=4; L3:=5; "END";
1127      "FOR" L:=L1,L2,L3 "DO""BEGIN" S:=S+1;  J:=II+S;
1128                        LD[J]:=DEF[I+L]* 10.0*20;
1129               J:=(T+T-J)*(J-1)"DIV"2+J;
1130               MX[J]:=10.0*20;
1131               "END";
1132         "END";
1133         "END" OF RESIN2;
1134         "PROCEDURE" DISPLAY2(FORCE,FLR,B);
1135         "VALUE" FLR,B; "INTEGER" FLR,B; "ARRAY" FORCE;
1136         "BEGIN""INTEGER" I,J,Q,L,TY;
1137          ALIGNED(6,4); DIGITS(6);
1138    "PRINT"'('F'');
1139         "IF" B=0 "THEN""BEGIN"
1140         "PRINT"'('L'',SAMELINE,'FLOOR ',FLR,' DISPLACEMENTS,',''L'');
1141         "PRINT"'('L'',SAMELINE,'MESH NODE'S4'VERT.DIS'S6'XROT'S8'YROT');
1142         "END""ELSE""BEGIN"
1143         "PRINT"'('L'',SAMELINE,'SHEAR DISPLACEMENTS: FLOOR',FLR,''L'');
1144    "PRINT"'('L'',SAMELINE,'MESH NODE'S4' ZROT'S8'X-DISP'S8'Y-DISP');
1145         "END";
1146               TY:=TYPE[FLR];      N:=EPF[TY,1];
1147         "FOR" I:=1 "STEP" 1 "UNTIL" N  "DO"
1148         "BEGIN""PRINT"'('L'',SAMELINE,!);
1149                     Q:=(I-1)*3;
1150         "FOR" J:=1,2,3 "DO""PRINT" SAMELINE,FORCE[Q+J];
1151         "END";
1152         "END" OF DISPLAY2;
1153          FLR:=0;
1154    RETN: FLR:=FLR+1;
1155          TY:=TYPE[FLR];
1156          MM:=3*EPF[TY,1];
1157          P:=MM;
1158          KK:=MM*(MM+1)"DIV"2;
1159         "BEGIN""REAL""ARRAY" FLORMX[1:KK],FLORLD,FORCE[1:P];
1160    "INTEGER""ARRAY" ROW[1:EPF[TY,1]];
1161    "FOR" I:=1"STEP"1"UNTIL"EPF[TY,1]"DO" ROW[I]:=I;
1162         "FOR" I:=1 "STEP" 1 "UNTIL" KK "DO" FLORMX[I]:=0;
1163         "FOR" I:=1 "STEP" 1 "UNTIL" P  "DO" FORCE[I]:=FLORLD[I]:=0;
1164         "BEGIN""INTEGER" K,L,EL;
1165    "REAL" A,B,D,PP,AA,BB,U,YM,T;"REAL""ARRAY" ES[1:12,1:12];
1166          AA:=BB:=0;
1167          U:=LOAD[TY];
1168         "FOR" EL:=1 "STEP" 1 "UNTIL" EPF[TY,3] "DO"
1169         "BEGIN"  I:=SCON[EL,TY,1];  K:=SCON[EL,TY,3];
1170                  J:=SCON[EL,TY,2];  L:=SCON[EL,TY,4];
1171     A:=XS[TY,J]-XS[TY,I];       B:=YS[TY,I]-YS[TY,K];
1172               A:=ABS(A); B:=ABS(B);
1173      YM:=PROP[TY,1];   PP:=PROP[TY,2];     T:=PROP[TY,3];
1174        D:=(YM*T*T*T)/(12*(1-PP*PP));
1175         "IF" AA"NE"A "OR" BB"NE"B "THEN""BEGIN"   FINITE(A,B,D,PP,ES);
1176                                                   AA:=A;  BB:=B; "END";
1177               ASSEMBLE(I,J,K,L,P,FLORMX,ES);
1178               FORMLOAD(FLORLD,U,I,J,K,L,A,B);
1179              "END";
1180    ADDBEAM(TY,12*NODE,E,G,0,FLORMX,ROW,100);
```

```
1181     FLOFIX(TY,EPF[TY,4],LINK,0,3*EPF[TY,1],ROW,FLORMX);
1182        RESIN2(DEF,FLORMX,FLORLD,FLR,0);
1183          ELIM(FLORMX,FLORLD,P,P);
1184          MILE(FLORMX,FORCE,FLORLD,P,P);
1185        DISPLAY2(FORCE,FLR,0);
1186      GAMMA(FORCE,FLR,XS,YS,SCON,PROP,EPF);
1187    ADDBEAM(FLR,12*NODE,E,G,0,FORCE,ROW,1);
1188    "END";
1189         "FOR" I:=1 "STEP" 1 "UNTIL" KK "DO" FLORMX[I]:=0;
1190         "FOR" I:=1 "STEP" 1 "UNTIL" P  "DO" FORCE[I]:=FLORLD[I]:=0;
1191         "BEGIN""INTEGER" K,L,EL;
1192         "REAL" A,B,D,PP,AA,BB,T,FAC,YM;
1193         "REAL""ARRAY" ES[1:12,1:12];
1194          AA:=BB:=0;
1195         "FOR" EL:=1 "STEP" 1 "UNTIL" EPF[TY,3] "DO"
1196         "BEGIN"  I:=SCON[EL,TY,1];  J:=SCON[EL,TY,2];
1197                  K:=SCON[EL,TY,3];  L:=SCON[EL,TY,4];
1198              A:=XS[TY,J]-XS[TY,I];  B:=YS[TY,I]-YS[TY,K];
1199              A:=ABS(A);   B:=ABS(B);
1200         YM:=PROP[TY,1];  PP:=PROP[TY,2];   T:=PROP[TY,3];
1201              D:=(YM*T*T*T)/(12*(1-PP*PP));
1202              FAC:=0.25;
1203         "IF" AA"NE"A "OR" BB"NE"B "THEN""BEGIN"
1204                   STRESS(PP,YM,T,A,B,ES,FAC);
1205                    AA:=A;  BB:=B;  "END";
1206          ASSEMBLE(I,J,K,L,P,FLORMX,ES);
1207         "END";
1208              SHEARLOAD(FLORLD,ROW,TY,1,SHLD);
1209    ADDBEAM(TY,12*NODE,E,G,1,FLORMX,ROW,100);
1210     FLOFIX(TY,EPF[TY,4],LINK,1,3*EPF[TY,1],ROW,FLORMX);
1211                          RESIN2(DEF,FLORMX,FLORLD,FLR,1);
1212              ELIM(FLORMX,FLORLD,P,P);
1213              MILE(FLORMX,FORCE,FLORLD,P,P);
1214              DISPLAY2(FORCE,FLR,1);
1215    ADDBEAM(FLR,12*NODE,E,G,1,FORCE,ROW,1);
1216     "IF" FLR<FLOR "THEN""GOTO" RETN;
1217    "END";
1218    "END";
1219        COLUMN(DEF,CON,XC,YC,ZC,PAR);
1220    "END";
1221    "END";
1222    "END";
1223    "END";
```